Theoretical and Computational Fluid Dynamics manuscript No.
(will be inserted by the editor)

On identification of self-similar characteristics using the Tensor Train decomposition method with application to channel turbulence flow

Thomas von Larcher · **Rupert Klein**

Abstract A study on the application of the Tensor Train decomposition method to 3D direct numerical simulation data of channel turbulence flow is presented. The approach is validated with respect to compression rate and storage requirement. In tests with synthetic data, it is found that grid-aligned self-similar patterns are well captured, and also the application to non grid-aligned self-similarity yields satisfying results. It is observed that the shape of the input Tensor significantly affects the compression rate. Applied to data of channel turbulent flow, the Tensor Train format allows for surprisingly high compression rates whilst ensuring low relative errors.

Keywords self-similarity; turbulent flows; Tensor Train format

1 Introduction

In wall-bounded turbulent flows characteristic coherent patterns, heterogeneous structures limited in time appearing irregularly at varying positions, are observed in distinct spatial regions [24], [41]. A number of experimental and numerical studies were able to describe some (quasi-) coherent features rather accurately, e.g. quasi-streamwise vortical structures are identified as a form of quasi-coherent structures [3], [2]. Here, a primary focus of research is the near-wall region, i.e. $y^+ < 40$ (y^+ as wall coordinate), where low-speed streaks are observed moving away from the wall, and, consequently, a flow towards the wall is required for continuity reasons. Indeed, regions of such wall-directed rapidly moving fluid, called sweeps, are observed, e.g. [7], and it has been discussed that these patterns as well as the low-speed streaks positively contribute to the production of turbulent energy, e.g. [52], [53].

The spectrum of turbulent kinetic energy first outlined by Richardson, [40], reveals a cascade of scales in its inertial subrange where energy is transferred by self-similar eddies from larger scales to smaller and smallest scales. In that sense, kinetic energy is produced

Th. von Larcher
Freie Universität Berlin, Institute of Mathematics, Arnimallee 9, 14195 Berlin, Germany
Tel.: +49-30-838-55093
E-mail: larcher@math.fu-berlin.de

R. Klein
Freie Universität Berlin, Institute of Mathematics, Arnimallee 6, 14195 Berlin, Germany

in the energy-containing range at the largest scales and dissipated at the smallest scales by (molecular) viscosity. In the inertial subrange, however, kinetic energy is neither produced nor dissipated. Since the pioneering work by Kolmogorov, [26], considerable time and effort has been invested in the field of Large Eddy Simulation (LES) to model the sub-grid scale (hereafter SGS for short) stress Tensor such that the SGS model can act independently of the flow field. This approach is reasonable as the results by e.g., [27], [34], [29], imply that the turbulent motion is statistically isotropic and the spatial and geometrical information transferred from the large scales to the small scales is lost. Hence, in turbulent flows, eddies at smaller scales are universal. Recent high-resolution numerical simulations obviously confirm established theoretic approaches that turbulent flows contain hierarchies of self-similar structures like vortex tubes or sheet-like flow patterns, e.g. [5].

Today, designing LES closures to model the unresolved small scales beyond statistical correlations, e.g. [45], is still subject of intense research. One branch follows the idea to reconstruct the fluctuations of the unresolved scales, that is, information of the dynamical, time-dependent, turbulent flow field is used. [1] develops a stochastic approach of an adaptive deconvolution model for the subgrid scale closure; [16] suggests a variational finite element formulation; [20] develops a variational multiscale method where adaption controls the influence of an eddy viscosity model. Self-similarity, however, is not resolved per-se in multiscale models. For this purpose, wavelet decomposition methods have been applied to data of turbulent flows, e.g. [8], and they were recently used to split the coherent signal from the incoherent part, e.g. [22], [43]. Similarity models, e.g. [30], and fractal models, e.g. [44], are promising tools as they extrapolate self-similarity to the small scales.

Tensor product decomposition methods, [13], were originally developed to yield low-rank, i.e. data-sparse, representations or approximations of high-dimensional data in mathematical issues. Multidimensional data sets, data of dimension 3 or higher, require massive storage capacity that strongly depends on the (Tensor) dimension, d, and on the number of entities per dimension, n. The datasize or storage requirement scales with $\mathcal{O}(n^d), n = \max_i\{n_i\}$. It is that curse of dimensionality that makes it difficult to handle higher-order Tensors or big data in an appropriate manner. It has been shown that Tensor product decomposition methods are as good as approximations by classical functions, e.g., like polynomials and wavelets, and that they allow very compact representations of large-data sets. Novel developments focus on hierarchical Tensor formats as e.g. the Tree-Tucker format, [11], and the Tensor Train format, [36], [12]. Nowadays, those hierarchical methods are successfully applied in e.g., physics or chemistry, where they are used in many body problems and quantum states.

Our study is concerned with the question of whether Tensor decomposition methods can support the development of improved understanding and quantitative characterisation of multiscale behavior of turbulent flows, cf. e.g. [46]. In the past, various Tensor decomposition methods were developed, e.g., the r-term or Canonical format and the Tucker decomposition, e.g. [47]. More recently, hierarchical formats as e.g., the hierarchical Tucker format [11], became more prominent due to the multiscale nature of problems in mathematics and physics. We refer to e.g. [25] and [10] for surveys of low-rank approximation techniques and specifically Tensor approximation formats.

Here, we apply the hierarchical Tensor Train format to data of a 3D direct numerical simulation (DNS) of a turbulent channel flow. We aim at capturing self-similar structures that might be hidden in the data. However, as those multiscale flow structures in highly irregular flows are not commonly aligned with the underlying grid but are translated, stretched, and rotated, we, at first, use synthetic data to evaluate the suitability of the method to generally detect self-similarity.

Provided that our tests yield promising results, those quantitative features could be helpful in developing a LES closure approach based on and extending the idea of fractal or dynamic SGS models, e.g. [9], [42]. Therefore, if proved positively, a long-term goal would be the construction of a self-consistent closure for LES of turbulent flows that explicitly exploits the Tensor decomposition approach's capability of capturing self-similar structures. Our approach is automatically linked with the following questions: (i) Can real data from multiscale dynamics be approximated or represented by the Tensor Train decomposition technique and how compact are the resulting storage schemes, i.e what compression rate can be achieved at which level of accuracy? (ii) Does the Tensor Train approximated data retain the dynamics? (iii) Is the Tensor Train format suitable for detecting cascades of scales in real data and in turbulence data in particular?

The article is organized as follows. The Tensor Train format used here is introduced in the next section, sect. 2. Results of tests with some synthetic data series and application to data of channel turbulence flow are presented in section 3 and discussed in section 4. The paper then ends with concluding remarks, section 5.

2 The Tensor Train format

In this section, we introduce the Tensor Train format, that is the Tensor decomposition method of choice for our study. As tensor decomposition techniques are based on the higher order generalization of the matrix singular value decomposition (hereafter referred to as HOSVD for short) we give a short recapitulation of the matrix SVD first. A description of the HOSVD basic operation is then given within the description of the Tensor Train decomposition.

SVD decomposes a matrix into a product of matrices, which then represents the original matrix: $\mathsf{A} = \mathsf{U}\mathsf{S}\mathsf{V}^{\mathsf{T}}$, where $\mathsf{A} \in \mathbb{R}^{m\times n}$ is the original matrix, $\mathsf{U} \in \mathbb{R}^{m\times m}$ and $\mathsf{V} \in \mathbb{R}^{n\times n}$ are orthonormal matrices and $\mathsf{S} \in \mathbb{R}^{m\times n}$ is a diagonal matrix with the singular values, σ_n, of A on its diagonal. Note that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$ and the number of non zero singular values is equal to the rank of A: $rank(\mathsf{A}) = r$. If the spectrum of the singular values contains only a few large entries and a sharp cut-off to the remaining tail, considerable data compression rates can be realized by truncating the matrices to the significant part of the spectrum. E.g., an approximation of the original matrix, A, with the first three singular values reads $\mathsf{A} \approx \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \sigma_3 u_3 v_3^T$. Moreover, a truncation to rank r results in a storage requirement of $mr + r + nr$ instead of mn for the original matrix A.

In the Tucker format a Tensor is decomposed into a product of a core Tensor, G, and factor matrices, U_i. Representation of a d-dimensional Tensor $\mathsf{X} \in \mathbb{R}^{n_1\times\ldots\times n_d}$ in the Tucker format reads

$$\mathsf{X}(n_1,\ldots,n_d) = \sum_{k_1,\ldots,k_d} \mathsf{G}(k_1,\ldots,k_d)\ \mathsf{U}_1(n_1,k_1)\ \mathsf{U}_2(n_2,k_2)\ \ldots\ \mathsf{U}_\mathsf{d}(n_d,k_d). \tag{1}$$

Note that the rank indices k_i vary in the range $k_i = 1,\ldots,r_i$. $\mathsf{G} \in \mathbb{R}^{r_1\times\ldots\times r_d}$ is a Tensor of order d and its entries imply the level of interaction between the components, and $\mathsf{U}_\mathsf{i} \in \mathbb{R}^{n_i\times r_i}$. The Tucker format uses the advantages of the HOSVD with a closed set of low-rank Tensors. The main disadvantage of Tucker decompositions is the high storage requirement as the core Tensor takes r^d elements, r as rank of the cores, resulting in a total storage requirement of $\mathscr{O}(dnr + r^d)$. Furthermore, Tucker decompositions are not unique.

The Tensor Train format is an extension of the Tucker decomposition. The key idea of Tensor Train decomposition is to separate a high-dimensional Tensor into its component

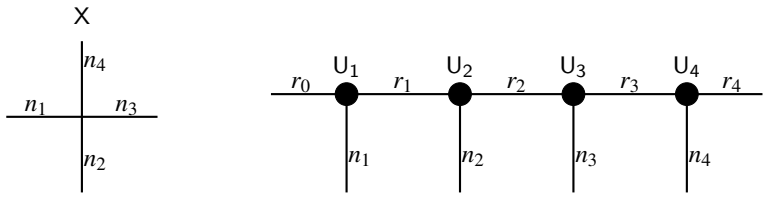

Fig. 1 Decomposition of a 4th-order Tensor $\mathsf{X}(n_1,n_2,n_3,n_4) \in \mathbb{R}$ into the Tensor Train format. Left: sketch of the input Tensor. Right: sketch of the Tensor Train format with U_i $(i = 1,\dots,4)$ as component Tensors, $\mathsf{U}_1 \in \mathbb{R}^{n_1 \times r_1}$, $\mathsf{U}_\mathsf{i} \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$, for $i = 2,3$, and $\mathsf{U}_4 \in \mathbb{R}^{r_3 \times n_4}$. Note that $r_0 = r_4 = 1$ by definition.

Tensors (modes) by maintaining the advantages of the Tucker format but avoiding the exponentially increase in storage requirement of the core Tensor. In the Tensor Train decomposition, the modes are then Tensors of order 3 by construction, and the format is designed such that the storage requirement scales linearly with order d: $\mathscr{O}(dnr^2)$. Previous studies show that the Tensor Train format allows for a much higher attainable compression rate compared to the low-rank approximation formats mentioned in the beginning of the section, e.g. [37].

In extension to the Tucker format, the representation of a d-dimensional Tensor X in the Tensor Train format reads

$$\mathsf{X}(n_1,\dots,n_d) = \sum_{k_1=1}^{r_1} \dots \sum_{k_{d-1}=1}^{r_{d-1}} \mathsf{U}_1(n_1,k_1)\, \mathsf{U}_2(k_1,n_2,k_2)\, \dots\, \mathsf{U}_\mathsf{d}(k_{d-1},n_d). \tag{2}$$

The component Tensors U_i are obtained by the step-by-step application of the matrix SVD. The procedure is as follows: in the first step, the d-dimensional Tensor X is reshaped into a 2-dimensional matrix $\mathsf{A}_1 \in \mathbb{R}^{n_1 \times n_2 \dots n_d}$ to which a SVD is applied, $\mathsf{A}_1 = \mathsf{U}_1\mathsf{S}_1\mathsf{V}_1^\mathsf{T}$, and the first mode $\mathsf{U}_1 \in \mathbb{R}^{n_1 \times r_1}$ is obtained. The remaining matrices of the SVD are contracted to one matrix $\mathsf{A}_2 = (\mathsf{S}_1\mathsf{V}_1^\mathsf{T}) \in \mathbb{R}^{r_1 \times n_2 \dots n_d}$ which is used for the second step. In the second step, A_2 is reshaped, at first, so that $\mathsf{A}_2 \in \mathbb{R}^{r_1 n_2 \times n_3 \dots n_d}$. Then, a SVD is applied, again, with $\mathsf{A}_2 = \mathsf{U}_2\mathsf{S}_2\mathsf{V}_2^\mathsf{T}$, and the second mode $\mathsf{U}_2 \in \mathbb{R}^{r_2 \times n_3}$ is obtained. By following this procedure successively, the remaining modes $\mathsf{U}_3,\dots,U_d$ are obtained. Note, that we need to apply the SVD d-times in total. As an example, figure 1 shows a graphical representation of the resulting Tensor Train network for an input Tensor of order 4.

The Tensor Train decomposition reveals some important topics: (a) in the Tensor Train network, the different modes U_i are linked by the ranks r_i (as sketched in figure 1). Thus, in the sense of the procedure just described parameter r_1 in the first step directly affects subsequent steps. (b) the parameters r_i determine the approximation quality. In case of Tensor approximation, as opposed to its representation, the r_i are limited to some maximum rank, r, and a lower value of r usually results in lower storage requirement for the approximated Tensor but also results in decreased approximation quality. The ranks r_i, therefore, are also called compression ranks or TT-ranks (TT as Tensor Train). This is because the value of r determines the number of rows to be kept for the next step and sets the cut-off of the matrices A_i. The hope is, that the omitted rows will be of small norm and the truncated component Tensors then will represent the significant part of the input Tensor. However, the optimal value of r is not known a priori and it is not guaranteed that the omitted rows are indeed of small norm. That implies consequences for the application of the Tensor Train decomposition to real data as we will see later.

3 Results

Here, we present the results of the Tensor Train approximation applied to data. We use the open source Tensor library *Xerus* developed at the Technical University Berlin, Germany, [15]. The relative error of the approximation with respect to the original Tensor is measured in the Frobenius norm that, for a d-dimensional Tensor X, reads

$$||\mathsf{X}|| = \sqrt{\sum_{n_1=1}^{N_1}\sum_{n_2=1}^{N_2}\cdots\sum_{n_d=1}^{N_d} x_{n_1\cdots n_d}^2}, \tag{3}$$

and the relative error in the Frobenius norm then reads

$$e = \frac{||(\mathsf{Y}-\mathsf{X})||}{||\mathsf{X}||}, \tag{4}$$

with Y as approximation of X.

We start with some tests with synthetic data series to demonstrate principals of the Tensor Train format and to debate the suitability of the method to detect self-similar structures. We, also, illustrate its capability to detect both, grid-aligned and non-grid-aligned self-similar patterns. Since real data are usually superimposed with noise we test the method on identifying self-similar structures in noisy data, too. Finally, the Tensor Train decomposition is applied to data of a high Reynolds number channel turbulence flow.

3.1 Principal operation of the Tensor Train approximation

We compute a data series of a function, $f(x)$, that involves an overlay of sine-functions of different periods (figure 2). The function reads

$$f(x) = 3.125\sin(x)\sin(2x)\sin(4x), \qquad x \in [0, 2\pi] \tag{5}$$

Here, $f(x)$ is sampled with $2^{14} = 16384$ data points. By construction, $f(x)$ involves similarity in the sense that $f_1 = f(x)$ for $x \in [0,\pi]$ and $f_2 = f(x)$ for $x \in [\pi, 2\pi]$ are symmetric to the zero line of the abscissa, i.e. $f_1 = -f_2$. But, $f(x)$ does not involve self-similarity in the proper sense.

Formally, the data series is a Tensor of order 1, i.e. a vector, and $\mathsf{X}(n_1) \in \mathbb{R}$ with $n_1 = 1,\ldots,2^{14}$. Without loss of generality, we can reshape the 1d-Tensor into a Tensor of order 3, that is $\mathsf{T}(n_1,n_2,n_3) \in \mathbb{R}$ with $\mathsf{T}[2,2,4096]$. Note, that the choice of the shape of the Tensor T is initially arbitrary but has a significant impact on the result as we will see below.

Applying the Tensor Train decomposition to T, it turns out that the input Tensor is approximated exactly, i.e. represented, for TT-rank ≥ 2, and we find $r_1 = 1$ and $r_2 = 2$. Figure 3a shows a sketch of the approximated Tensor written in the Tensor Train format. The modes U_1 and U_2 read

$$\mathsf{U}_1 = \frac{1}{\sqrt{2}}\begin{pmatrix}-1 & 1\end{pmatrix} \qquad \mathsf{U}_2 = \frac{1}{\sqrt{2}}\begin{pmatrix}-1 & 1\\ 1 & 1\end{pmatrix},$$

and the basis is given by the components of mode $\mathsf{U}_3 \in \mathbb{R}^{2\times 4096}$ (see figure 3b). The storage of T in the Tensor Train format requires 8200 entries, i.e., about half the storage requirement in the original Tensor notation. Note that, in the limit $r_k = 1\ \forall k$, the Tensor approximation yields a relative error $e = 0.33$ with a storage requirement of 4102 entries.

Table 1 Relative error, e, and storage requirement for approximation of Tensor $\mathsf{T}_2[2,2,2,2048]$ for TT-ranks 1 to 4.

TT-rank	e	storage requirement
1	0.448	2056
2	0.047	4112
3	0.004	6164
4	1.8e-15	8216

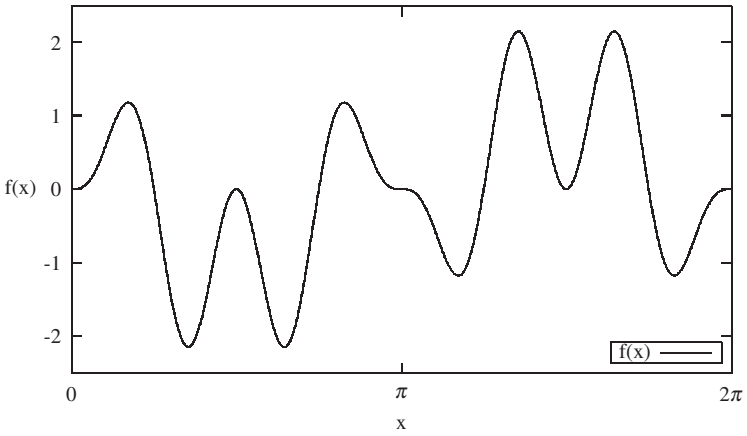

Fig. 2 Graph of $f(x) = 3.125\sin(x)\sin(2x)\sin(4x)$, $x \in [0, 2\pi]$.

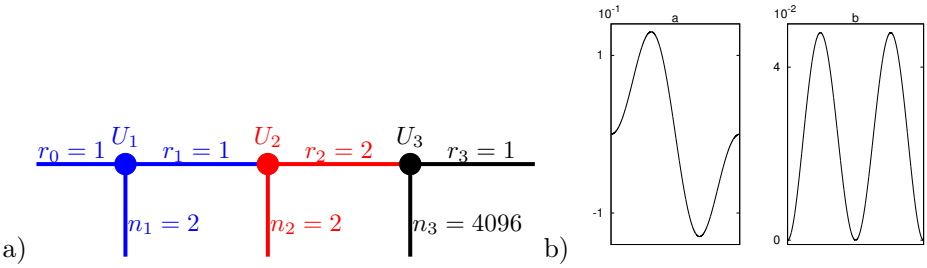

Fig. 3 Left: schematic view of the Tensor $\mathsf{T}[2,2,4096]$ written in the Tensor Train format. Note $r_0 = r_3 = 1$ by definition. Colors denote the 3rd order Tensors $\mathsf{U}_1, \mathsf{U}_2, \mathsf{U}_3$. Right: the two basis vectors of U_3.

We, now, perform a test on the sensitivity of the Tensor Train decomposition on the shape of the input Tensor. Suppose, we reshape the given data series, (5), into a Tensor T_2 of order 4 where $\mathsf{T}_2[2,2,2,2048]$. Applying the Tensor Train approach to T_2, we find that TT-rank 4 is needed as minimum TT-rank to represent it and table 1 shows the relative error and the storage requirement for TT-rank 1 to 4. Compared to the previous findings, here, using TT-rank 2 now approximates the input Tensor T_2 rather than represent it, as T in the former example, and the relative error is about 0.05.

The bottom line of the two tests presented here, i.e., approximation of Tensor T and T_2, is that Tensor Train decomposition appears highly sensitive to the shape of the input Tensor.

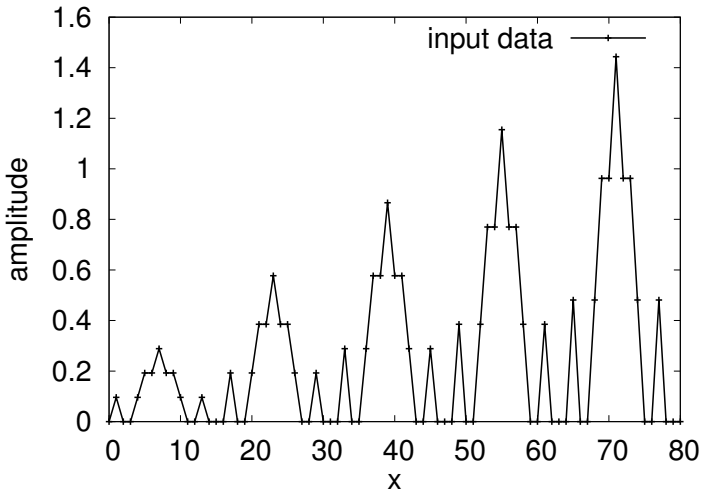

Fig. 4 Sequence of the data series with self-similar triangular patterns. The self-similar structure is a block of 16 data points that is repeated and scaled-up. Note that the data series involves 1024 entries in total.

Table 2 Minimum TT-ranks and storage requirement to represent different shapes of the input Tensor T_3. Note that T_3 is a Tensor of order 9 in all cases.

T_3	TT-ranks $(r_1,\ldots,r_8)$	storage requirement
$[2,2,2,2,2,2,2,2,4]$	2,2,2,2,2,1,2,3	70
$[2,2,2,2,2,2,2,4,2]$	2,2,2,2,2,1,2,2	66
$[2,2,2,2,2,2,4,2,2]$	2,2,2,2,2,1,3,2	70
$[2,2,2,2,2,4,2,2,2]$	2,2,2,2,2,2,3,2	82
$[2,2,2,2,4,2,2,2,2]$	2,2,2,2,1,2,3,2	70

3.2 Detection of self-similar structures

Now, we extent the first example and test the Tensor Train format to detect self-similar structures hidden in data series. For this purpose, we compute a data series comprising $2^{10} = 1024$ data points in total that consists of a triangular structure of $2^4 = 16$ data points which is not only repeated but also scaled-up in every loop, see figure 4. Thus, it is in line with the power of two sequence. As shown previously, the impact of the shape of the input Tensor on the resulting storage requirement can be significant. Therefore, we reshape the data series in a Tensor of order 9 where each of the 9 dimensions have 2 entries except the second to last one that has 4 entries, i.e. $\mathsf{T}_3[2,2,2,2,2,2,2,4,2]$. This means the self-similar structure in the data series is maintained and, furthermore, remains grid-aligned and is not disturbed by splitting of the Tensor. Decomposing the input Tensor into the Tensor Train format, we find TT-rank 2 as minimum rank to represent it, and the storage requirement, 66 entries ($\approx 6.4\,\%$ of the original Tensor notation), is very low. Variations of the input Tensor's shape results in very low storage requirements, too, but it turns out that the shape of T_3 demands the lowest storage requirement and the lowest TT-rank. Table 2 shows results for varieties of the input Tensor entries.

Finally, we consider the properties of the Tensor Train format in case the input Tensor involves self-similar structures not aligned with the grid, that is, the self-similar patterns are

not in line with the power of two sequence. For that purpose we compute a top-hat function where the squares are scaled up, see figure 5a. The data series contains $2^{14} = 16384$ entries in total. To test the effect of noisy data, we perform two additional tests and add random noise of different amplitude, here amplitude factor 1 and 10, to the data series. Before we apply the Tensor Train decomposition we reshape the data series into a Tensor of order 14, thus each dimension has 2 entries, $\mathsf{T}_4[2,2,2,2,2,2,2,2,2,2,2,2,2,2]$. With this arbitrary choice, we intentionally neglect our a priori knowledge on the character of the data series. This is reasonable as we can not expect detailed knowledge of the data structure in case of real data.

Figure 5b shows the storage requirement (hereafter also denoted as datasize) in the Tensor Train format against the relative error for a number of TT-ranks for both, the data series without noise and with noise amplitude factor 1 and 10. Approximation of Tensor T_4 at TT-rank 1 yields the lowest storage requirement but the largest relative error. In case of no noise T_4 is represented at TT-rank 5 where the relative error is about 10^{-15}, and storage in the Tensor Train format requires about 500 entries thus $\approx 3.0\,\%$ of the original datasize. Limiting the TT-rank to 2 (4) yields an approximation error of about $0.2\,\%$ ($0.02\,\%$), and storage of the approximated Tensor in the Tensor Train format requires about 100 (350) entries, i.e. $0.6\,\%$ ($2.1\,\%$) of the datasize in original Tensor notation. Approximation of the noisy data series for TT-rank 5 to about 80 shows a drastic increase in the storage requirement whilst only small changes in the relative error. Obviously, this is due to the small-scale noise the approximation of which naturally requires many more data entries.

Both data series, with noise amplitude 1 and with noise amplitude 10, are approximated exactly at TT-rank 128 at which the relative error drops to 10^{-15}. In both cases, the storage requirement of the representations in the Tensor Train format (43690 entries) substantially exceeds the storage requirement of the source data in the original Tensor format (16384 entries). For both noisy data series, we find the original datasize at TT-rank 45 where $e \approx 0.03$ (noise factor 10) and $e \approx 0.003$ (noise factor 1), resp. Furthermore, for a given datasize, the relative error strongly depends on the noise amplitude as long as the noise is not approximated well.

Interestingly, also for the noisy data series the relative error agrees well with that of the no-noise case up to TT-rank 4, TT-rank 3 for noise amplitude 10, indicating that the self-similar top-hat structure, large-scaled relative to the small scale noise, but not the noise itself is represented by low TT-ranks. Figure 5c shows the euclidian norm of the difference between the original Tensor and the approximated Tensor. At low TT-ranks, the euclidian norm agrees well in all cases underlining the previous statement that large-scale structures are resolved at low TT-ranks. At large TT-ranks, the graphs of both noisy data series show a significant drop at TT-rank 127 specifying the exact approximation of the input Tensor.

Figure 5d confirms that picture since the self-similar top-hat structure but not the noisy part (small-scaled with respect to the top-hat structure) is approximated well at TT-rank 4 (except the small-scale top-hats at the beginning of the data series). With TT-rank 2 as maximum permitted TT-rank, the approximation of the top-hat structures is less accurate but the general trend is obtained, at least. That result is of course linked with the mathematics of the algorithm discussed previously since the given TT-rank is linked with the number of rows, i.e. number of singular values, to be kept in the Tensor Train decomposition procedure (cf. section 2). Low rank, therefore, means a loss of information particularly about small-scales, that is usually hidden in higher eigenvalues of the matrices. In other words, large-scale structures are already being represented whilst small-scale patterns are still coarsely approximated.

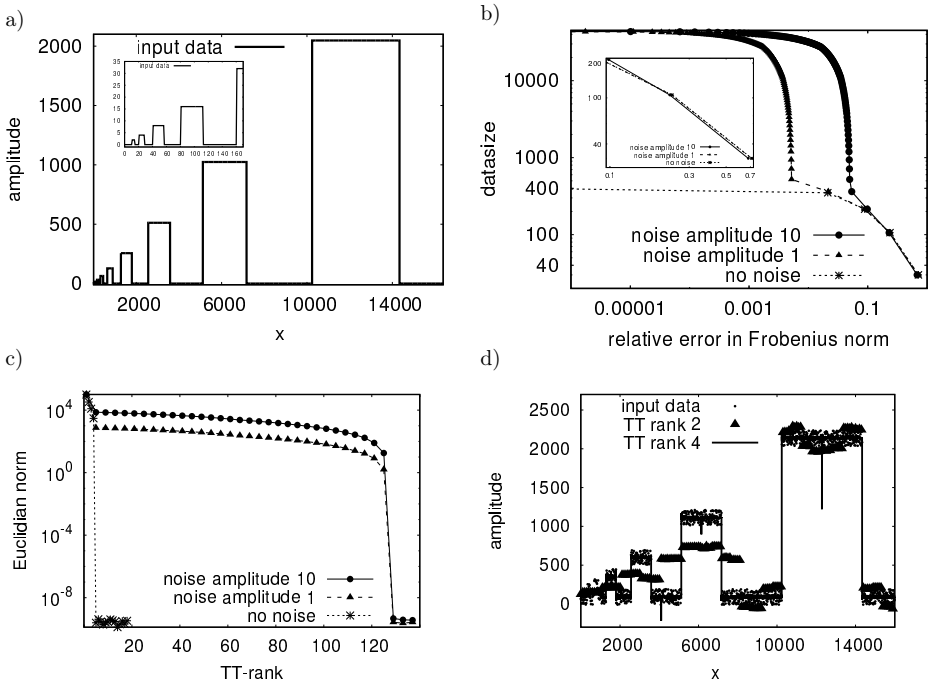

Fig. 5 Test of the top-hat function. a) synthetic data series, noise amplitude 0. b) storage requirement against relative error for series of TT-ranks. No noise: TT-rank $1,\ldots,4$; noise factor 1 and 10: TT-rank $1,2,\ldots,126$. c) Euclidian norm of the difference between original and approximated data series against TT-rank. d) data series of noise amplitude 10 for approximation with TT-rank 2 and TT-rank 4. Note that only every 14th entry of the original input data is plotted due to avoid overloading the figure.

In summary, the tests with different synthetic data series show that approximation in the Tensor Train format is efficient and promising in detecting self-similar patterns in particular if those structures are grid-aligned. Furthermore, we found that the minimal TT-rank to represent a data series in the Tensor Train format is sensitive to the shape of the input Tensor. The latter statement has been also discussed by [4].

Even if the self-similar structures are not grid-aligned, approximation in the Tensor Train format yields satisfying results particularly for data series without noise. Especially, the results yielded in the top-hat function test show that a straightforward truncation after a few TT-ranks, i.e. considering a finite number of singular values only, reflects some type of a low-pass filter.

3.3 Channel turbulence flow

3.3.1 Description of the data

We use data of 3D direct numerical simulation (DNS) of an isothermal and incompressible plane channel flow, figure 6a, computed by [48] using a pseudo-spectral Fourier-Chebyshev method. We, also, refer to [49], [50] for an in-depth description of data generation. Periodic boundary conditions are applied in the streamwise and spanwise direction and a no-slip boundary condition is set at both side walls of the wall-normal direction. In wall-normal

direction a Chebyshev-tau polynomial formulation has been used, and in streamwise and spanwise direction a Fourier representation has been formulated, cf. [19], [23], [33]. The box size in streamwise, wall-normal, and spanwise direction is $L_x = 2\pi$, $L_y = 2$, and $L_z = \pi$. The original grid spatial resolution is $600 \times 385 \times 600$ in (x,y,z).

The data are generated at friction-based Reynolds number $Re_\tau = 590$, the viscous length-scale is calculated to $\delta_\nu = 0.0017$. We apply the Tensor Train decomposition to velocity data (fig. 6b) and also to data of vorticity magnitude (fig. 6c) since in turbulent flows vorticity plays a key role in general and in wall turbulence in particular, and its time-dependent dynamics determine the stability of coherent structures. The vorticity data are generated by using the function *curl* of the MATLAB software package.

The scalar Q-criterion proposed by [17], [18] is a measure to demarcate vortex structures within turbulent flows. Figure 6d shows an iso-surface map of the Q-criterion at a given snapshot. Here, vortex tubes of different size and shape, rotated and stretched, can be identified indicating the highly turbulent flow.

In case of velocity data analysis the datasize is 380160000 ($\mathcal{O}(10^9)$) entries per snapshot. Data of vorticity magnitude demands a storage requirement of 126720000 ($\mathcal{O}(10^8)$) entries per snapshot.

In the following analysis, we use post-processed data, so-called fine grid (fgv hereafter) data, that is, the DNS velocity data have been re-calculated to obtain an equidistant grid increment in wall-normal direction, too. We refer to [28] for a detailed description of the post-processing algorithms. The grid size of the fgv data is $600 \times 352 \times 600$ in (x,y,z) and the grid increment in wall units is $\Delta x^+ = 6.1$, $\Delta y^+ = 3.4$, $\Delta z^+ = 3.0$.

Figure 7 shows calculated turbulent flow statistics for which 236 time steps of DNS and fgv data are used. No significant loss of information is recognized due to the fine grid post-processing procedure compared to the signatures of the original DNS grid data. However, the resolution of near-wall regions is slightly reduced in the fgv data due to underlying interpolation routines in the post-processing procedure. In particular, the viscous sublayer ($y^+ < 5$) is not well resolved anymore.

3.3.2 Application to velocity data

Since the tests with synthetic data series already have revealed the sensitivity to the shape of the input Tensor, see the previous subsection, we consider two different approaches for the analysis of the velocity data. Firstly, the input Tensor retains the original data structure, that is, it is a 4th-order Tensor $\mathsf{T}[600,352,600,3]$ where the last dimension contains the components of velocity. This approach is referred to as TT-approximation hereafter. On the other hand, we decompose every dimension into its prime factors, resulting in a Tensor of order 19

$$\mathsf{T}[n_1,\ldots,n_{19}] = \mathsf{T}[2,2,2,3,5,5,\ 2,2,2,2,2,11,\ 2,2,2,3,5,5,\ 3], \tag{6}$$

and apply the Tensor Train decomposition to this Tensor. The latter approach is referred to as QTT-approximation hereafter as it is similar to the Quantics Tensor Train (QTT) approach, [21], [35]. However, in the original QTT-approach, input Tensors are reshaped into pure binary representation.

For both approaches, figure 8 shows the storage requirement in the Tensor Train format against the relative error for various TT-ranks. Both, QTT- and TT-approximation, shows a significant increase in storage requirement if the TT-rank increases. Obviously, the QTT-approach yields less relative error with significantly less storage requirement. For example,

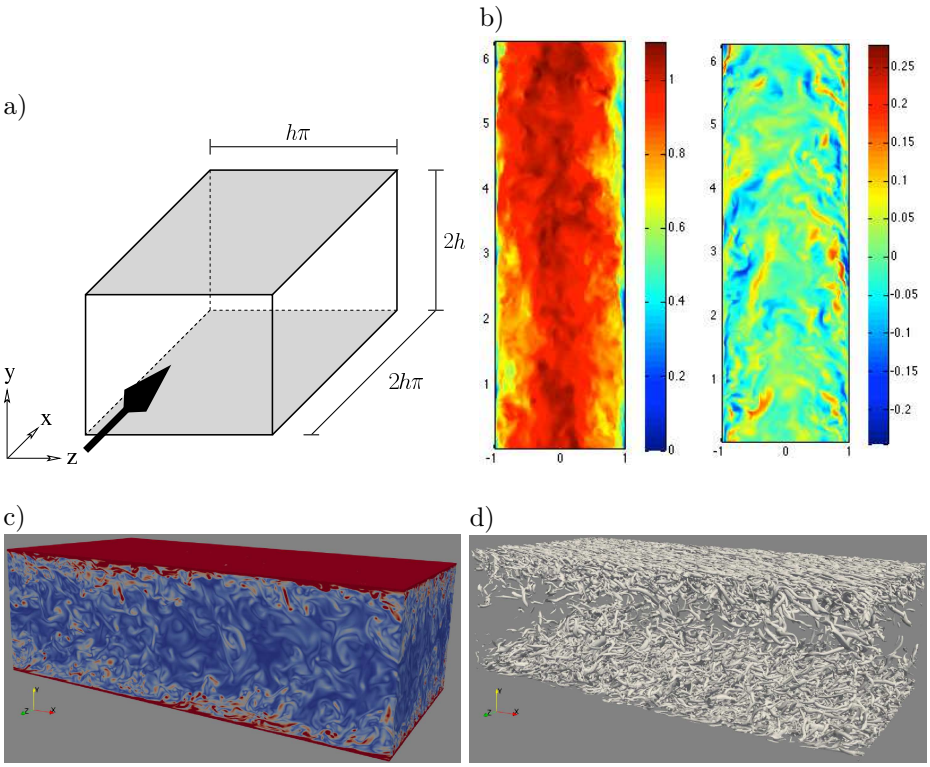

Fig. 6 Turbulent channel flow. a) sketch of the channel geometry (courtesy of M. Waidmann) with streamwise direction x, $L_x = 2h\pi$, spanwise direction z, $L_z = h\pi$, and wall-normal direction y, $L_y = 2h$, where $h = 1$. b) (x,y)-slice of the streamwise (u) velocity component (left panel) and of the spanwise (w) velocity component (right panel). Note that the main flow direction is from bottom to top. c) vorticity magnitude, linear scaling from 0 (blue) to 8 (red). d) iso-surface map of the Q-criterion (q=4).

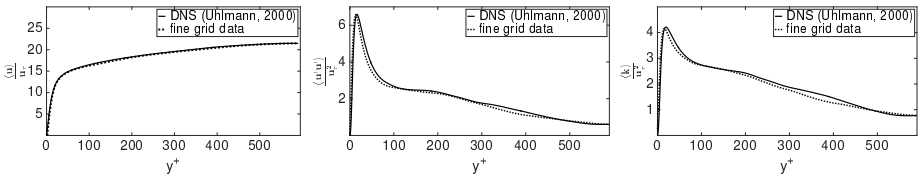

Fig. 7 Wall-normal profiles of DNS and fgv turbulent flow statistics normalized by friction velocity u_τ against wall unit y^+. Left panel: mean velocity profile $\langle u \rangle$, middle panel: Reynolds stress $\langle u'u' \rangle$, right panel: turbulent kinetic energy $\langle k \rangle$. Note that the channel center is at $y^+ = 590$.

given a relative error of 0.085 the QTT-approximation demands a storage requirement of about 18000 entries but TT-approximation demands about 60000 entries. Note that, induced by the different shapes of the input Tensors, the QTT-approach necessitates much larger TT-ranks at a given storage requirement.

Figure 9 for the TT-approach and figure 10 for the QTT-approach displays 2D-slices of the approximated velocity data for selected TT-ranks. Figure 9 shows images of velocity magnitude calculated from the TT-approximated velocity field. In addition, 1D cross-sections extracted in streamwise direction are displayed. Also, figure 10 shows the magni-

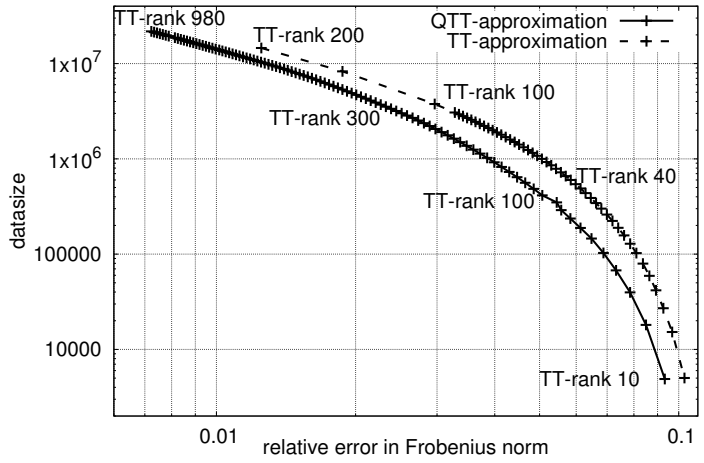

Fig. 8 Approximation of velocity data. Storage requirement against relative error, e, for QTT- (solid line) and TT-approximation (dashed line) for various TT-ranks. Note the different TT-ranks in the QTT- and TT-approximation. For TT-approximation, the increment is set to 2 up to TT-rank 100 and to 50 up to TT-rank 200 afterwards. For QTT-approximation, the increment is set to 10 up to TT-rank 980. Note that the labels on the left hand side of the curves as well as the label *TT-rank 980* are associated with the QTT-curve and the labels on the right hand side of the curves are associated with the TT-curve.

tude of velocity for the QTT-approach. Especially at lower TT-ranks, block-like patterns are evident in the wall-normal y-direction, e.g., figure 10a. They result from the split of the input Tensor in the QTT-approach, that is, the dimension in y-direction is separated in blocks of 11 entries each. Similarly, the streamwise and spanwise direction is split in blocks of 2 entries each and that might also be visible in figure 10a. Both, the TT- and the QTT-approach, exhibit a general feature, that is, we find a smooth reflection of the turbulent flow at lower TT-ranks, and an increase in the TT-rank results in enhanced resolution of small-scale patterns accordingly.

Furthermore, figure 9e illustrates a cross-section in wall-normal direction extracted for the TT-rank 2 approximation shown in panel 9a. Interestingly, that cross-section involves the box-shaped time-space mean velocity profile typical found in channel turbulence flows along the wall-normal direction, cf. e.g. [39]. In that sense, TT-approximation constrained to TT-rank 2 acts as an averaging process apparently.

Now, we are interested in identifying those regions in the channel flow that show the most significant deviations from the original in the approximated data. In both cases, TT-approximation and QTT-approximation, reconstruction of the velocity field for higher TT-ranks, for example TT-rank 100 in figure 9c and 10b, already shows several details of small-scale features of the turbulent flow by eye, even if the relative error is significant. For the QTT-approximation, figure 10e for TT-rank 20 and 10f for TT-rank 100 shows the difference between the approximated and the original data. Obviously, significant deviations are located towards the near-wall region while the flow interior appears well resolved already at lower TT-ranks. This picture is becoming more and more visible with increasing TT-ranks where details of the small-scale features are getting resolved.

The previous statement indicates that at a given TT-rank the flow interior is approximated with greater accuracy than the near-wall region. To verify this assumption we ex-

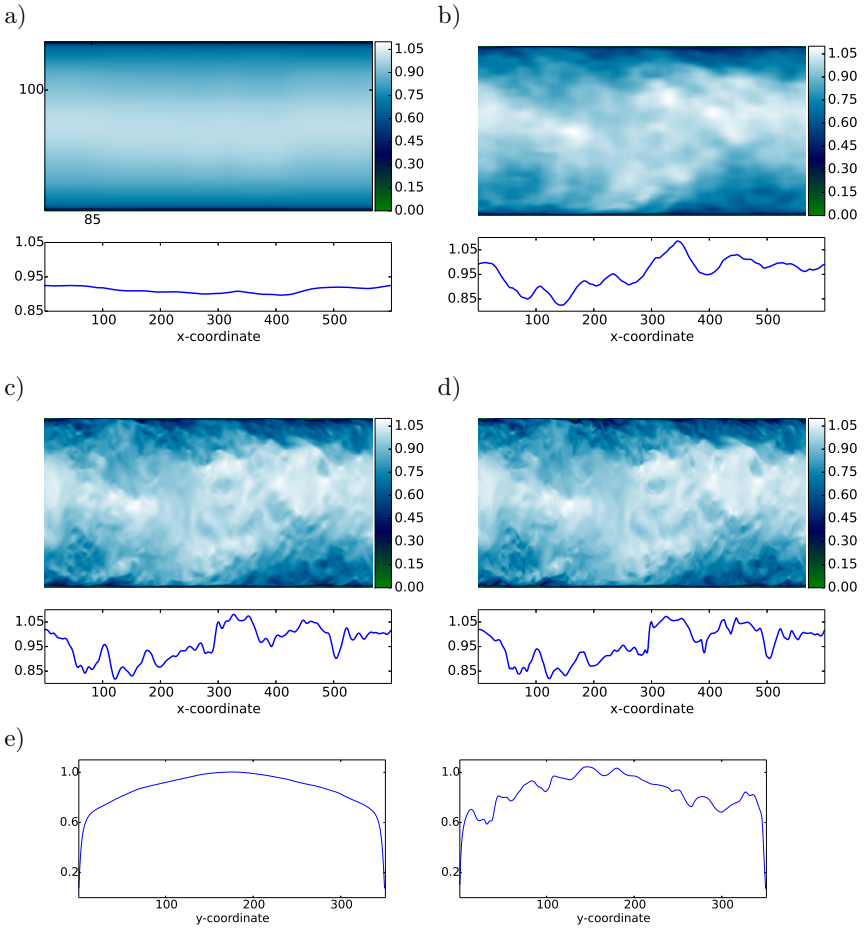

Fig. 9 Approximation of the velocity field in the TT-approach. a)-d): (x,y)-slices of velocity magnitude at $z = 300$ (mid-channel) and 1d cross-sections in streamwise direction extracted at $(y,z) = (100,300)$; a) for TT-rank 2 ($e \approx 0.103$), b) for TT-rank 22 ($e \approx 0.072$), c) for TT-rank 100 ($e \approx 0.030$), d) original data. e) cross-section in the wall-normal direction extracted at $(x,z) = (85,300)$; TT-rank 2 (left panel) and original data (right panel). The appropriate coordinate $x = 85$ as well as $y = 100$ is marked in panel a).

tract (x,z)-slices in the near-wall region as well as in the flow interior and apply the QTT-approximation to the data. Here, the input Tensor is of order 13 as we split both, the x- and z-velocity component, in 6 dimensions each as described in (6), and one dimension of the input Tensor comprises the velocity components. Thus, the input Tensor has 1080000 data entries in total. The results of the QTT-approximation, figure 11, support the previous statement as the relative error of the approximation at the flow center is much smaller than that at the near-wall regions. Interestingly, the approximation yields similar results for the $y^+ = 13.5$ case and for the $y^+ = 33.7$ case. The latter result may have its reason in the fact that both slices are located in the same wall layer that is the viscous wall region according to [39].

To estimate the likeness of the QTT-approximation also quantitatively, we follow Reynolds decomposition of the velocity field $\mathbf{U} = (u,v,w)$ and determine its fluctuating part. In gen-

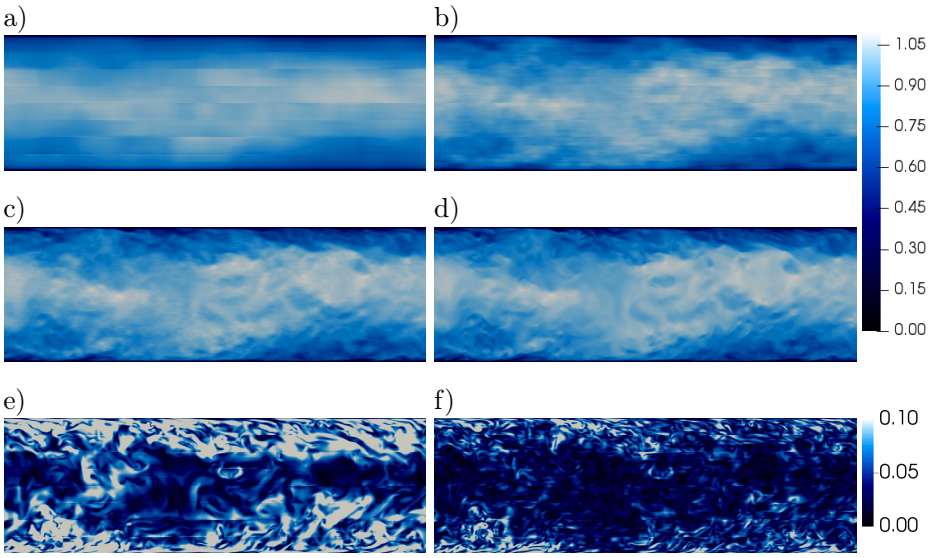

Fig. 10 Approximation of the velocity field in the QTT-approach. (x,y)-slice of velocity magnitude extracted at $z = 300$ (mid-channel). a) for TT-rank 20 ($e \approx 0.085$), b) for TT-rank 100 ($e \approx 0.055$), c) for TT-rank 420 ($e \approx 0.020$), d) original data; e) difference between approximated data and original data for TT-rank 20. f) as e) but for TT-rank 100.

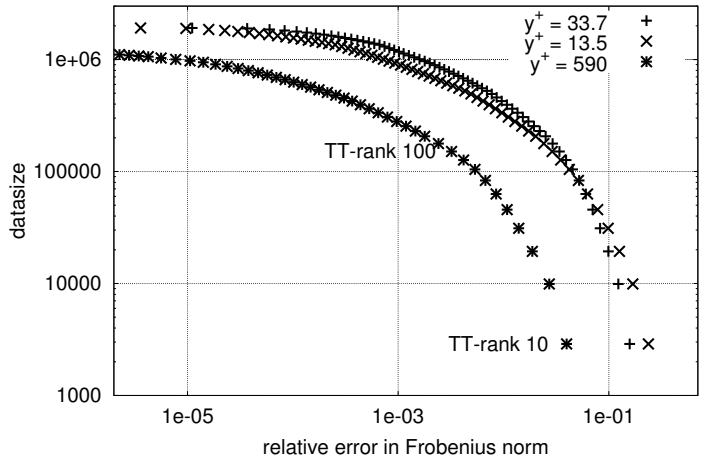

Fig. 11 QTT-approximation for (x,z)-slices extracted at different positions in wall-normal direction. At $y^+ = 590$ (mid-channel): star symbols, at $y^+ = 13.5$: cross symbols, at $y^+ = 33.7$: plus symbols. Increment of the TT-ranks: 10. Note that, at a given TT-rank, the storage requirement is independent of the data series due to the same shape of the input Tensors (e.g., about 1e+06 at TT-rank 370).

Table 3 QTT-approximation of velocity data. Spatial mean of the normal stresses of the Reynolds-stress Tensor at a given time step for different TT-ranks. ( ) denotes the proportion to the reference value [%].

TT-rank	$\langle u'^2 \rangle$	$\langle v'^2 \rangle$	$\langle w'^2 \rangle$
20	0.0287	0.00016	0.00060
	(91.4)	(9.4)	(25.0)
100	0.0303	0.0011	0.0016
	(96.5)	(64.7)	(66.7)
420	0.0312	0.0016	0.0023
	(99.4)	(94.1)	(95.8)

eral, the fluctuating part of the velocity vector, $\mathbf{U}' = (u', v', w')$, reads

$$\mathbf{U_i}' = \mathbf{U_i} - \langle \mathbf{U_i} \rangle, \quad i = 1,2,3 \tag{7}$$

with $\langle \mathbf{U_i} \rangle$ as the spatial, time or ensemble mean of the appropriate velocity component. Since we, here, are considering turbulence data at a given time step rather than a time sequence of snapshots, we determine both, $\langle \mathbf{U_i} \rangle$ and the normal stresses, $\langle u'^2 \rangle$, $\langle v'^2 \rangle$, $\langle w'^2 \rangle$, of the Reynolds-stress Tensor by spatial averaging. For the fgv data, we find $\langle u'^2 \rangle = 0.0314$, $\langle v'^2 \rangle = 0.0017$, $\langle w'^2 \rangle = 0.0024$. For the approximated data, we find good agreement with these reference values particularly for $\langle u'^2 \rangle$ already at TT-rank 100 ($e = 0.055$), see table 3.

We verify the apparent good agreement of the approximated data also at lower TT-ranks by calculating turbulent flow statistics for time series of QTT-approximated data for which again 236 snapshots are used, cf. figure 12. The mean velocity profile underlines the previous observations that the streamwise velocity component is approximated well already at lower TT-ranks. However, the diagram of the Reynolds stress as well as that of the turbulent kinetic energy (TKE), figure 12b and 12c, reveals differences between the fgv data and the approximated data at TT-rank 100. Approximations using TT-rank 320 or higher TT-ranks then show good agreement also for Reynolds stress and for TKE. Here, the undulating patterns, particularly visible in the TT-rank 100 case, result from the block-like structure of the input Tensor, as already described previously.

In summary, we can say that approximation of the 3D velocity field with the Tensor Train decomposition yields significant reduction in storage requirement and low relative errors already at lower TT-ranks. The QTT-approach, that is, the split of the components of the input Tensor into its prime factors before applying the Tensor Train decomposition, leads to a further reduction in storage requirement with respect to the TT-approach for which the structure of the input Tensor remains as given in the fgv data. In the QTT-approach, the calculated turbulent flow statistics show that wall-normal profiles are not approximated well for the TT-rank 100 case but that they are for TT-rank 320 or higher.

3.3.3 Application to vorticity data

We conclude this section with the application of the Tensor Train decomposition to data of vorticity magnitude. Here, we follow two different approaches. First, we calculate the vorticity magnitude from the already QTT- and TT-approximated velocity data. Secondly, we compute the magnitude of vorticity from the fgv velocity field and apply the QTT-approximation afterwards. For the latter, we split the data of vorticity magnitude into its prime factors analogous to the QTT-approach of the velocity data analysis. Thus, the shape of the input Tensor

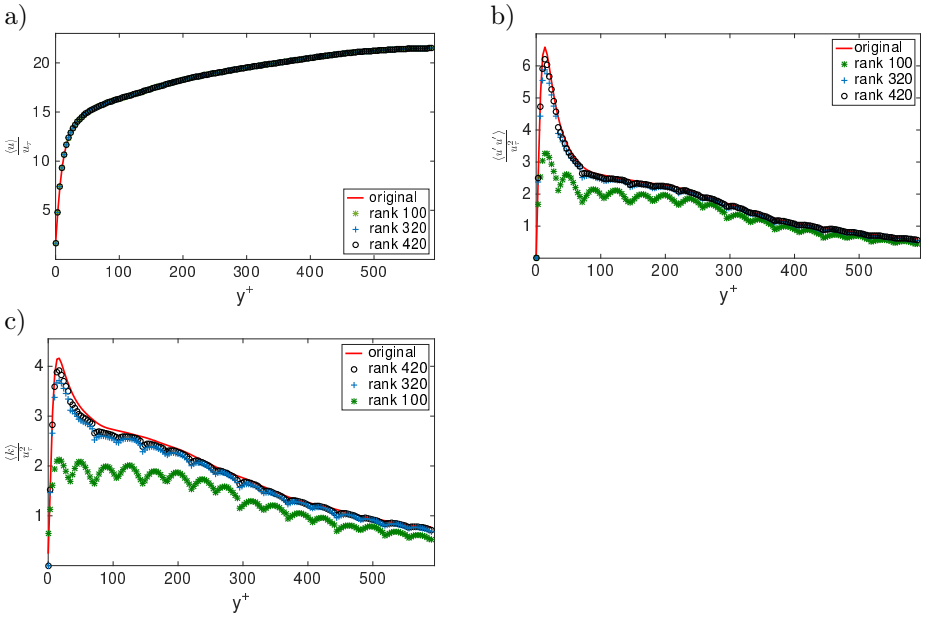

Fig. 12 QTT-approximation of velocity data: wall-normal profiles of turbulent flow statistics normalized by friction velocity u_τ against wall unit y^+ for fgv (original) data and for data of specific TT-rank approximations (TT-rank 100, 320 and 420). a) mean profile of streamwise velocity component $\langle u \rangle$, b) Reynolds stress $\langle u'u' \rangle$, c) turbulent kinetic energy $\langle k \rangle$.

is almost identical to the velocity data analysis but now contains the magnitude of vorticity instead of three velocity components.

For the first approach, figure 13 shows results for QTT- and TT-approximated data. Similar to the velocity data analysis, also the vorticity data reveal small scale features at higher TT-ranks, and approximation at TT-rank 420 for the QTT-approach, panel 13b, and at TT-rank 100 for the TT-approach, panel 13e, shows satisfying results at least qualitatively.

For the second approach, figure 14 shows results for the storage requirement against the relative error. Generally, the trend of a decrease in the relative error and accompanying increase in the TT-rank is similar to what we have found in the analysis of the velocity data but the magnitude of the relative error is about an order higher, cf. figure 8 for the velocity data. That result is reasonable as vorticity dominates at small scales in contrast to velocity. Thus, compared to the analysis of the velocity data a large relative error is not surprising in case of approximation of vorticity data at low TT-ranks.

4 Discussion

In the previous section, we observed that the Tensor Train decomposition method yields high compression rates whilst ensuring low relative errors not only in the tests applying synthetic data series but also for the channel turbulence flow. The efficiency of the Tensor Train format in data compression becomes particularly evident in the tests with synthetic data where very high compression rates are achieved. Regarding the turbulent cannel flow data, even if the magnitude of the relative error is much larger in the vorticity data analysis compared to

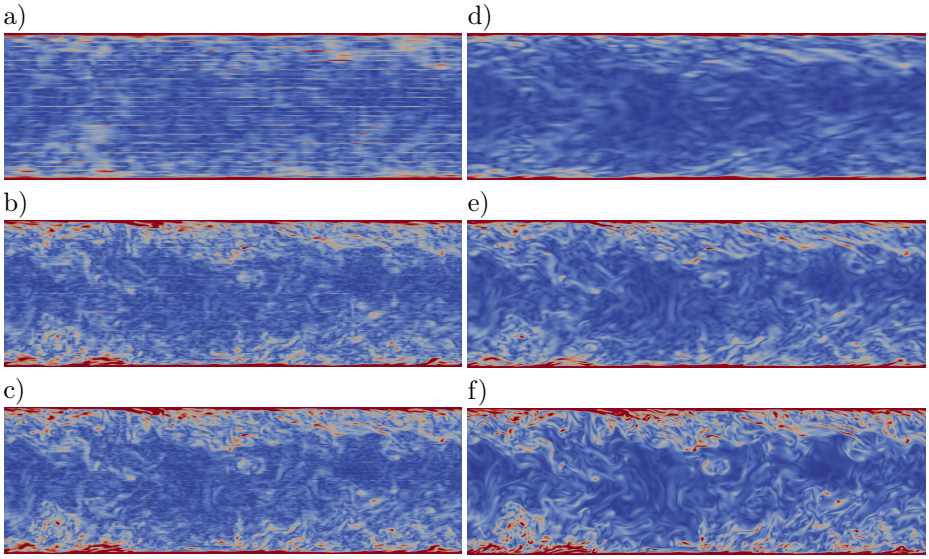

Fig. 13 Vorticity magnitude calculated from QTT- and TT-approximated velocity data. (x,y)-slices extracted at mid-channel. Left column: QTT-approximation for a) TT-rank 100 ($e \approx 0.054$), b) TT-rank 420 ($e \approx 0.020$), and c) TT-rank 600 ($e \approx 0.014$). Right column: TT-approximation for d) TT-rank 40 ($e \approx 0.057$) and e) TT-rank 100 ($e \approx 0.030$). f) original data. Note that the colorbar scale is the same in all panels, that is, it is a linear scale from 0 (blue) to 8 (red).

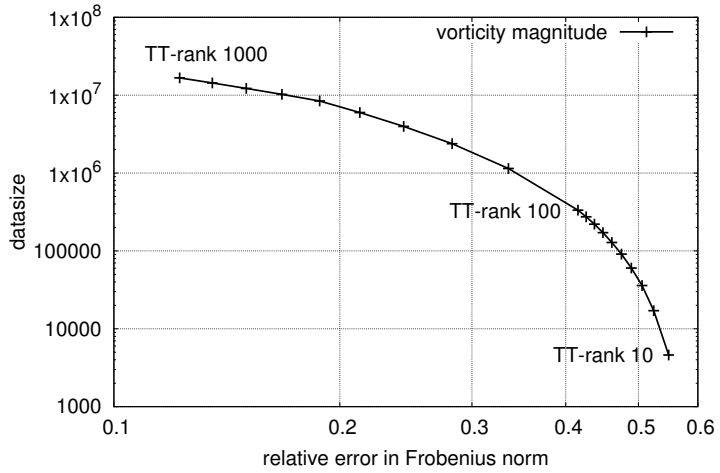

Fig. 14 QTT-approximation of data of vorticity magnitude. Storage requirement against relative error calculated for TT-rank 10 to 1000. Rank increment 10 up to TT-rank 100 and 100 up to TT-rank 1000.

approximation of velocity data, transforming the data in the Tensor Train format leads to a significant reduction in storage requirement for both, velocity and vorticity data.

To put the Tensor Train format into perspective, we compare our results with results of a wavelet approximation as wavelets are known being appropriate tools to reveal self-similarities in fractal functions, e.g., [6], [51], [31]. Furthermore, the similarity between Tensor Train decomposition and wavelet transforms has been recently described by [38] who showed that Tensor Train decomposition can be considered as an algebraic wavelet transform with adaptively determined filters.

Here, we apply a wavelet analysis to the same synthetic data series used in section 3.2. For this purpose, we employ the GNU Scientific Library software and exploit the frequently used Daubechies orthogonal wavelet family with $k/2$ vanishing moments where we set $k = 4$, 6, and 16. Loosely speaking, the larger $k/2$, the more signal components can be encoded. For example, a $k = 6$ wavelet transform encodes constant, linear, and quadratic signal components. Besides the Daubechies wavelet family we also use the Haar wavelet (with $k = 2$ by construction). Again, we calculate the error norm of the approximated data with (4) to compare the relative error of the wavelet approximation with the results of the Tensor Train approach.

At first, we consider the data series of self-similar triangular patterns that involves 1024 entries in total (cf. figure 4). In section 3.2, we demonstrated that the input Tensor is represented in the Tensor Train format taking 66 entries in the most favourable scenario, and 82 entries are needed in a worst-optimal split of the input Tensor, see table 2. The results of the wavelet analysis using different number of components of its transform, *nc*, are displayed in table 4. Here, considering 66 components, the Daubechies 6 wavelet reveals the minimal error ($e \approx 0.4$). Naturally, increasing the number of components ensures a simultaneous increase in the approximation quality and for $nc = 800$, i.e. about 78 % of the data, the Haar wavelet approximates the data series with greatest success ($e \approx 0.005$). We conclude that the Haar wavelet transform and also our choice of Daubechies wavelets is apparently not able to represent that specific data series using a small number of components only, and the latter might be due to the shape of their scale functions and wavelet functions with respect to the triangular shape of the self-similar patterns.

Now, we revisit the noise-free sequence of the top-hat data series of non-grid aligned squares (cf. figure 5a). We remind the reader, that the data series is represented in the Tensor Train format at TT-rank ≥ 5 and its storage then requires 492 entries. A priori, the Haar wavelet appears well suited by construction to represent the data series requiring only a small number of components of its transform. Indeed, we find that the data series is represented using 177 components of the Haar wavelet transform ($e \approx 10^{-16}$). Further tests considering the same number of components of the Daubechies 4 wavelet transform show that the data series is approximated rather than represented and the relative error is of the order of 10^{-3}.

The comparison with the wavelet analysis, on the one hand, point out the efficiency of the Tensor Train format in detecting and representing self-similar patterns hidden in data series. On the other hand, however, these tests underline the known result that wavelet transforms are able to attack self-similarity with great success. Since the results of the wavelet analysis of the synthetic data series are promising, too, it appears worth testing a wavelet transform to data of the channel turbulence flow. We, therefore, deploy the Daubechies 4 wavelet transform to vorticity magnitude (see section 3.3.3). Table 5 shows results for different number of components, *nc*, of the wavelet transform, and for comparison, the result of the QTT-approximation at appropriate TT-ranks, that is the storage requirement in the QTT-approach is equal to *nc*, is given, too. Obviously, the QTT-approach results in higher accuracy (lower relative error) particularly at lower TT-ranks. At higher TT-ranks, i.e., in

Table 4 Relative error, e, of wavelet approximation of the synthetic data series of triangular patterns. k as the wavelet index number, nc as number of components of the wavelet transform used for the approximation.

type k	nc	e
Daubechies 4	66	0.409
Daubechies 6	66	0.396
Daubechies 16	66	0.565
Haar 2	66	0.507
Daubechies 4	800	0.010
Daubechies 6	800	0.018
Daubechies 16	800	0.012
Haar 2	800	0.005

Table 5 Wavelet analysis of vorticity magnitude. Relative error, e, of wavelet approximation, nc as number of components of the wavelet transform used for the approximation.

Daubechies 4		QTT-approximation	
nc	e	TT-rank	e
91062	0.55	50	0.47
1143837	0.41	200	0.34
10233837	0.15	700	0.17

terms of the wavelet analysis considering a large number of components for the approximation, both methods reveal comparable accuracy.

5 Concluding remarks

In this article, the Tensor Train decomposition technique, a specific branch of the family of Tensor product decomposition methods, has been applied to data of a numerical simulation of channel turbulence flow. Both, velocity and vorticity data were used to test the robustness of the approximation procedure.

Our study is concerned with the detection of self-similar patterns in turbulent flow data. We, firstly, analyzed synthetic data to demonstrate the Tensor Train decomposition‘s capability to capture self-similarities. The synthetic data tests with and without noise provide promising results. Grid aligned self-similarity is well captured and non grid-aligned self-similarity is approximated at low TT-ranks strengthening the suitability of the method. Moreover, with no noise, non grid-aligned self-similar structures are approximated exactly at low TT-ranks resulting in extreme data compression ratios.

Applied to channel turbulence data, especially the QTT-approach, that is in our definition decomposing the entries of each dimension of the input Tensor into its prime factors, allows for surprisingly high compression rates whilst ensuring low relative errors. For example, QTT-approximation at TT-rank 100 results in a compression factor of roughly 1000 and in a relative error of about 5 %. At TT-rank 2000, the relative error is about 0.2 % with a compression factor of about 5, that is a storage requirement of $\approx 10^8$ (instead of $\approx 10^9$ for the original data). Realizing that low-rank approximations in the TT- and QTT-approach amount to some kind of averaging, it is reasonable that the analysis of vorticity data show a relative error much higher than for velocity data. Vorticity data have a lot more significant structure in the small scales, and, therefore, averaging is expected to be generally bad.

However, our results demonstrate that, apparently, low-rank representation of highly irregular and chaotic flows can not be expected. The latter result is reasonable as in turbulent flow multiscale and self-similar structures are stretched, rotated, and translated, and, naturally, the underlying algorithm of the Tensor Train format has reasoned limitations in capturing those deformed structures on different scales efficiently. Here, we re-emphasize the statement made previously that the Tensor Train format obviously acts as some type of low-pass filter in the sense that a straightforward truncation after a small number of TT-ranks reflected the space mean of the 3D velocity field of the turbulent channel flow. We, therefore, conclude that modifications of the Tensor Train decomposition method are needed if one is interested in self-similarity in scale, including associated amplitude scalings as we expect them in turbulence data.

The overall aim of our ongoing study is to identify characteristic self-similar patterns in multiscale flows, thus coherent structures in turbulent flows, that appear repeatedly on different scales. In future work, we will use and combine further multiscale data analysis techniques such as, e.g., wavelets, shearlets and turbulent detection methods. In particular combination of advanced data analysis and machine learning techniques as, e.g., the FEM-VARX method by [14], [32], envisage encouraging new options, and our study presented here show that extension of those methods to tensor-valued data would provide promising opportunities. The characterization of self-similar structures would then provide the basis for extrapolating ensembles of such flow structures simulated on the coarse LES-grid in scale to generate a self-consistent model of unresolved fluctuations.

Acknowledgements This research has been funded by Deutsche Forschungsgemeinschaft (DFG) through grant CRC 1114 'Scaling Cascades in Complex Systems', Project B04 'Multiscale Tensor decomposition methods for partial differential equations'. The authors thank Prof. Illia Horenko (CRC 1114 Mercator Fellow) as well as Prof. Reinhold Schneider and Prof. Harry Yserentant for rich discussions and for steady support. Thomas von Larcher thanks Sebastian Wolf and Benjamin Huber (both at TU Berlin, Germany) very much for developing the Tensor library *xerus* which has been used for data analysis, as well as for their round the clock support in the project. The data were generated and processed using resources of the North-German Supercomputing Alliance (HLRN), Germany, and of the Department of Mathematics and Computer Science, Freie Universität Berlin, Germany. The authors thank Alexander Kuhn and Christian Hege (both at Zuse Institute Berlin, Germany) for steady support in data processing and data visualisation.

References

1. Adams, N.: A stochastic extension of the approximate deconvolution model. Phys. Fluids **23(055103)**, 1–9 (2011)
2. Adrian, R., Meinhart, C., Tomkins, C.: Vortex organization in the outer region of the turbulent boundary layer. J. Fluid Mech. **422**, 154 (2000). DOI 10.1017/S0022112000001580
3. Bakewell, H., Lumley, J.: Viscous sublayer and adjacent wall region in turbulent pipe flow. Phys. Fluids **10**(9), 1880–1889 (1967). DOI 10.1063/1.1762382
4. Ballani, J., Grasedyck, L.: Tree adaptive approximation in the hierarchical tensor format. Siam J. Sci. Comput. **36**, A1415–A1431 (2014)
5. Bürger, K., et al.: Vortices within vortices: hierarchical nature of vortex tubes in turbulence. arXiv:1210.3325 [physics.flu-dyn] (2013)
6. Cattani, C.: Wavelet analysis of self similar functions. Journal of Dynamical Systems and Geometric Theories **9**(1), 75–97 (2011). DOI 10.1080/1726037X.2011.10698594. URL http://dx.doi.org/10.1080/1726037X.2011.10698594
7. Corino, E., Brodkey, R.: A visual investigation of the wall region in turbulent flow. J. Fluid Mech. **37**, 1–30 (1969). DOI 10.1017/S0022112069000395
8. Fröhlich, J., Uhlmann, M.: Orthonormal polynomial wavelets on the interval and applications to the analysis of turbulent flow fields. SIAM J. Appl. Math. **63**(5), 1789–1830 (2003)
9. Germano, M., Piomelli, U., Moin, P., Cabot, W.H.: A dynamic subgrid scale eddy viscosity model. Phys Fluids **A 48**, 273–337 (1991)

10. Grasedyck, L., Kressner, D., Tobler, C.: A literature survey of low-rank tensor approximation techniques. GAMM-Mitteilungen **36**, 53–78 (2013)
11. Hackbusch, W., Kühn, S.: A new scheme for the tensor representation. J. Fourier Anal. Appl. **15**, 706–722 (2009)
12. Hackbusch, W., Schneider, R.: Tensor spaces and hierarchical tensor representations. In: S. Dahlke, W. Dahmen, M. Griebel, W. Hackbusch, K. Ritter, R. Schneider, C. Schwab, H. Yserentant (eds.) Extraction of quantifiable information from complex systems, *Lecture notes in computational science and engineering*, vol. 102, pp. 237–361. Springer, New York (2014)
13. Hitchcock, F.: The expression of a tensor or a polyadic as a sum of products. Journal of Mathematics and Physics **6**, 164–189 (1927)
14. Horenko, I.: On clustering of non-stationary meteorological time series. Dyn. Atmos. Ocean **49**, 164–187 (2010)
15. Huber, B., Wolf, S.: Xerus - a general purpose tensor library. https://libxerus.org/ (2014–2015)
16. Hughes, T.: Multiscale phenomena: Greens functions, the dirichlet-to-neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. Comput. Methods Appl. Mech. Engrg. **127**, 387–401 (1995)
17. Hunt, J.C.R.: Vorticity and vortex dynamics in complex turbulent flows. In: Canadian Society for Mechanical Engineering, Transactions (ISSN 0315-8977), vol. 11, no. 1, 1987, p. 21-35., vol. 11, pp. 21–35 (1987)
18. Hunt, J.C.R., Wray, A.A., Moin, P.: Eddies, streams, and convergence zones in turbulent flows. In: Studying Turbulence Using Numerical Simulation Databases, 2 (1988)
19. Jimenez, J., Moin, P.: The minimal flow unit in near-wall turbulence. J. Fluid Mech. **225**, 213–240 (1991)
20. John, V., Kindl, A.: A variational multiscale method for turbulent flow simulation with adaptive large scale space. J. Comput. Phys. **229**, 301–312 (2010)
21. Khoromskij, B.: O(d log n)-quantics approximation of n-d tensors in high-dimensional numerical modeling. Constr. Approx. **34**, 257–280 (2011)
22. Khujadze, G., Nguyen van yen, R., Schneider, K., Oberlack, M., Farge, M.: Coherent vorticity extraction in turbulent boundary layers using orthogonal wavelets. J. Physics: Conference Series **318(022011)**, 1–10 (2011)
23. Kim, J., Moin, P., Moser, R.: Turbulence statistics in fully developed channel flow at low Reynolds number. J. Fluid Mech. **177**, 133–166 (1987)
24. Kline, S., Robinson, S.: Quasi-coherent structures in the turbulent boundary layer. i - status report on a community-wide summary of the data. In: S. Kline, N. Afgan (eds.) Near-Wall Turbulence, pp. 200–217. Routledge (1990)
25. Kolda, T., Bader, B.: Tensor decompositions and applications. SIAM Review **51**, 455–500 (2009)
26. Kolmogorov, A.: The local structure of turbulence in incompressible viscous fluid for very large reynolds number. C.R. Acad. Sci. U.S.S.R. **30**, 301 (1941)
27. Kolmogorov, A.: A refinement of previous hypothesis concerning the local structure of turbulence in a viscous incompressible fluid at high reynolds number. J. Fluid Mech. **62**, 82 (1962)
28. von Larcher, T., Beck, A., Klein, R., Horenko, I., Metzner, P., Waidmann, M., Igdalov, D., Gassner, G., Munz, C.D.: Towards a framework for the stochastic modelling of subgrid scale fluxes for large eddy simulation. Meteorologische Zeitschrift **24**(3), 313–342 (2015). DOI 10.1127/metz/2015/0581. URL http://dx.doi.org/10.1127/metz/2015/0581
29. Lesieur, M.: Turbulence in Fluids: Stochastic and Numerical Modelling. Nijhoff (1987)
30. Liu, S., Meneveau, C., Katz, J.: On the properties of similarity subgrid-scale models as deduced from measurements in a turbulent jet. J. Fluid Mech. **275**, 83–91 (1994)
31. Mallat, S.: A Wavelet Tour of Signal Processing, third edn. Academic Press, Boston (2009)
32. Metzner, P., Putzig, L., Horenko, I.: Analysis of persistent non-stationary time series and applications. CAMCoS **7**, 175–229 (2012)
33. Moser, R., Kim, J., Mansour, N.: Direct numerical simulation of turbulent channel flow up to Re_τ=590. Phys. Fluids **11**(4), 943–945 (1999)
34. Oboukhov, A.: Some specific features of atmospheric turbulence. J. Fluid Mech. **62**, 77 (1962)
35. Oseledets, I.: Approximation of $2^d \times 2^d$ matrices using tensor decomposition. SIAM J. Matrix Anal. Appl. **31**, 2130–2145 (2010)
36. Oseledets, I., Tyrtyshnikov, E.: Breaking the curse of dimensionality, or how to use SVD in many dimensions. Siam J. Sci. Comput. **31**, 3744–3759 (2009)
37. Oseledets, I., Tyrtyshnikov, E.: TT-cross approximation for multidimensional arrays. Linear Algebra and its Applications **432**, 70–88 (2010)
38. Oseledets, I., Tyrtyshnikov, E.: Algebraic Wavelet Transform via Quantics Tensor Train decomposition. SIAM J. Sci. Comput. **33**, 1315–1328 (2011)
39. Pope, S.: Turbulent Flows. Cambridge University Press (2000)

40. Richardson, L.: Weather prediction by numerical process. Cambridge University Press (1922)
41. Robinson, S.: Coherent motions in the turbulent boundary layer. Ann. Rev. Fluid Mech. **23**, 601–639 (1991)
42. Sagaut, P.: Large Eddy Simulation for Incompressible Flows, 3 edn. Springer (2006)
43. Sakurai, T., Yoshimatsu, K., Schneider, K., Farge, M., Morishita, K., Ishihara, T.: Coherent structure extraction in turbulent channel flow using boundary adapted wavelets. J Turbul **18**, 352–372 (2017)
44. Scotti, A., Meneveau, C.: A fractal model for large eddy simulation of turbulent flows. Physica D **127**, 198–232 (1999)
45. Smagorinsky, J.: General circulation experiments with the primitive equations. Mon. Wea. Rev **93**, 99–164 (1963)
46. Ting, L., Klein, R., Knio, O.M.: Vortex dominated flows: analysis and computation for multiple scales, *Series in Applied Mathematical Sciences*, vol. 116. Springer Verlag, Berlin, Heidelberg, New York (2007)
47. Tucker, L.: Some mathematical notes on three-mode factor analysis. Psychometrika **31**, 279–311 (1966)
48. Uhlmann, M.: Generation of a temporally well-resolved sequence of snapshots of the flow-field in turbulent plane channel flow (2000). Published online: http://www-turbul.ifh.uni-karlsruhe.de/uhlmann/reports/produce.pdf (accessed at July 2017)
49. Uhlmann, M.: Generation of initial fields for channel flow investigation. Intermediate Report. (2000). Published online: http://www-turbul.ifh.uni-karlsruhe.de/uhlmann/home/report.html (accessed at July 2017)
50. Uhlmann, M.: The need for de-aliasing in a Chebyshev pseudo-spectral method. Technical Note No. 60. (2000). Published online: http://www-turbul.ifh.uni-karlsruhe.de/uhlmann/reports/dealias.pdf (accessed at July 2017)
51. Vergassola, M., Frisch, U.: Wavelet transforms of self-similar processes. Physica D: Nonlinear Phenomena **54**(1), 58–64 (1991). DOI https://doi.org/10.1016/0167-2789(91)90107-K. URL http://www.sciencedirect.com/science/article/pii/016727899190107K
52. Wallace, J., Eckelmann, H., Brodkey, R.: The wall region in turbulent shear flow. J. Fluid Mech. **54**(1), 3948 (1972). DOI 10.1017/S0022112072000515
53. Willmarth, W., Lu, S.: Structure of the Reynolds stress near the wall. J. Fluid Mech. **55**(1), 6592 (1972). DOI 10.1017/S002211207200165X