Communications on Applied Mathematics and Computation (2021) 3:257–279
https://doi.org/10.1007/s42967-020-00090-6

ORIGINAL PAPER

MERACLE: Constructive Layer-Wise Conversion of a Tensor Train into a MERA

Kim Batselier[1] · **Andrzej Cichocki**[2] · **Ngai Wong**[3]

Received: 20 December 2019 / Revised: 17 July 2020 / Accepted: 19 July 2020 / Published online: 19 October 2020

Abstract
In this article, two new algorithms are presented that convert a given data tensor train into either a Tucker decomposition with orthogonal matrix factors or a multi-scale entanglement renormalization ansatz (MERA). The Tucker core tensor is never explicitly computed but stored as a tensor train instead, resulting in both computationally and storage efficient algorithms. Both the multilinear Tucker-ranks as well as the MERA-ranks are automatically determined by the algorithm for a given upper bound on the relative approximation error. In addition, an iterative algorithm with low computational complexity based on solving an orthogonal Procrustes problem is proposed for the first time to retrieve optimal rank-lowering disentangler tensors, which are a crucial component in the construction of a low-rank MERA. Numerical experiments demonstrate the effectiveness of the proposed algorithms together with the potential storage benefit of a low-rank MERA over a tensor train.

Keywords Tensors · Tensor train · Tucker decomposition · HOSVD · MERA · Disentangler

Mathematics Subject Classification 15A23 · 15A69 · 65F99

1 Introduction

Tensor decompositions have played an important role over the past two decades in lifting the curse of dimensionality in a myriad of applications [3–5, 18, 26]. The key idea in lifting the curse of dimensionality with tensor decompositions is the usage of a low-rank approximation. Many kinds of decompositions have consequently been developed and each

This research was partially supported by the Ministry of Education and Science of the Russian Federation (grant 14.756.31.0001).

✉ Kim Batselier
k.batselier@tudelft.nl

[1] Delft Center for Systems and Control, Delft University of Technology, Delft, the Netherlands

[2] Skolkovo Institute of Science and Technology (SKOLTECH), 121205 Moscow, Russia

[3] The Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong, China

has its own rank definition. The canonical polyadic decomposition (CPD) [2, 15, 16] and the Tucker decomposition [2, 27] both generalize the notion of the matrix singular value decomposition (SVD) to higher-order tensors and have, therefore, received a lot of attention. More recent tensor decompositions are the tensor train (TT) [8, 9, 18, 21] and the hierarchical Tucker decomposition [12, 13]. It turns out that the latter two decompositions were already known in the quantum mechanics and condensed matter physics communities as the matrix product state (MPS) [23] and the tensor tree network (TTN) [25], respectively. The multi-scale entanglement renormalization ansatz (MERA) [10, 30] is an extension of the TTN decomposition, recently proposed in quantum mechanics but has so far not received enough attention in the numerical linear algebra community. A key component of the MERA is the so-called disentangler tensor, responsible for limiting the growth of the TTN-ranks over consecutive levels. Although the computation of a MERA from a given tensor can be deduced from [10], computations are intensive due to multiple contractions and do not allow for the discovery of optimal ranks of the decomposition. The contributions of this article address this area. Specifically, we

i) propose an algorithm that converts a given TT into a Tucker decomposition with guaranteed error bounds;
ii) propose an algorithm that converts a given TT into a MERA with guaranteed error bounds. This algorithm is called MERA constructive layer-wise expansion (MERACLE);
iii) propose an iterative algorithm that computes a rank-lowering disentangler.

The resulting ranks of the computed Tucker and MERA approximations are completely determined by a given upper bound on the relative approximation error. The conversion of a TT into a Tucker decomposition was first suggested in [7], where the corresponding algorithm uses an iterative alternating least squares (ALS) approach. It will be shown in this article that no ALS procedure is necessary. In fact, for a D-th order tensor it is sufficient to perform D consecutive SVD computations as described in Algorithm 1. It is then shown in Algorithm 2 that a TT can be converted into an L-layer MERA by applying Algorithm 1 $2L$ times. The obtained MERA ranks are, however, not optimal and this is identified to be due to the disentangler tensor computation. An iterative orthogonal Procrustes algorithm is proposed that to our knowledge for the first time ever, is able to compute optimal disentanglers that result in a minimal-rank MERA.

In Sect. 2 we introduce the notation and relevant tensor decompositions. The algorithm that converts a given TT into a Tucker decomposition with a guaranteed relative error bound is fully described in Sect. 3. The application of Algorithm 1 for the conversion of a given TT into a MERA with a guaranteed relative error bound is illustrated in Sect. 4. Section 5 discusses the problem of finding optimal disentangler tensors and the iterative Procrustes algorithm is proposed. Finally, in Sect. 6 numerical experiments demonstrate the effectiveness of the proposed algorithms.

2 Tensor Basics

A D-way or D-th order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ is a D-dimensional array where each entry is completely determined by D indices $i_1, \cdots, i_D$. The scalar D is also often called the order of the tensor. The convention $i_d = 1, 2, \cdots, I_d$ is used, together with MATLAB

colon notation. Boldface capital calligraphic letters $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \cdots$ are used to denote tensors, boldface capital letters $\boldsymbol{A}, \boldsymbol{B}, \cdots$ denote matrices, boldface letters $\boldsymbol{a}, \boldsymbol{b}, \cdots$ denote vectors, and Roman letters $a, b, \cdots$ denote scalars. The identity matrix of order N is denoted $\boldsymbol{I}_N$. The Frobenius norm $||\boldsymbol{\mathcal{A}}||_{\mathrm{F}}^2$ of a tensor $\boldsymbol{\mathcal{A}}$ is defined as the sum of squares of all tensor entries. The order of a tensor can be altered by grouping several indices together into a multi-index. The conversion of a multi-index $[i_1 i_2 \cdots i_D]$ into a linear index is per definition

$$[i_1 i_2 \cdots i_D] := i_1 + \sum_{k=2}^{D} (i_k - 1) \prod_{l=1}^{k-1} I_l. \tag{1}$$

In what follows, we will introduce three important tensor operations. The first tensor operation is the "reshape" operation, which changes the order of a given tensor and is commonly used to flatten tensors into matrices and vice versa.

Definition 1 The operator "reshape$(\boldsymbol{\mathcal{A}}, [J_1, J_2, \cdots, J_K])$" reshapes the d-way tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ into a tensor with dimensions $J_1 \times J_2 \times \cdots \times J_K$, with $\prod_{d=1}^{D} I_d = \prod_{k=1}^{K} J_k$.

Another important operation is the generalization of the matrix transpose to three or more indices.

Definition 2 The operator "permute$(\boldsymbol{\mathcal{A}}, \boldsymbol{p})$" rearranges the indices of $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ so that they are in the order specified by the vector $\boldsymbol{p}$. The resulting tensor has the same values of $\boldsymbol{\mathcal{A}}$ but the order of the subscripts needed to access any particular element is rearranged as specified by $\boldsymbol{p}$. All the elements of $\boldsymbol{p}$ must be unique, real, positive, integer values from 1 to D.

The definition of the "permute" operation allows one to write the transpose of a matrix $\boldsymbol{A}$ as permute$(\boldsymbol{A}, [2, 1])$. By combining both the reshape and permute operations, we can now introduce the mode-d matricization $\boldsymbol{A}_{<d>}$ of a tensor.

Definition 3 ([19, p. 459]) The mode-d matricization $\boldsymbol{A}_{<d>}$ of a D-way tensor $\boldsymbol{\mathcal{A}}$ is the matrix with elements

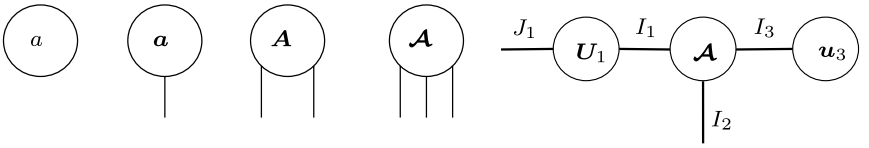
$$\boldsymbol{A}_{<d>}(i_d, [i_1 \cdots i_{d-1} i_{d+1} \cdots i_D]) := \boldsymbol{\mathcal{A}}(i_1, i_2, \cdots, i_D).$$

The mode-d matricization $\boldsymbol{A}_{<d>}$ is hence obtained from $\boldsymbol{\mathcal{A}}$ as

$$\boldsymbol{A}_{<d>} = \text{reshape}(\text{permute}(\boldsymbol{\mathcal{A}}, [d, 1, 2, \cdots, d-1, d+1, \cdots, D]), [I_d, I_1 \cdots I_D]).$$

The third and final important tensor operation is the summation over indices, also called contraction of indices. A particular common operation in this regard is the d-mode product of a tensor with a matrix.

Definition 4 ([19, p. 460]) The d-mode product, denoted $\boldsymbol{\mathcal{A}} \times_d \boldsymbol{U}_d$, of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_D}$ with a matrix $\boldsymbol{U}_d \in \mathbb{R}^{S_d \times I_d}$ is the tensor $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{I_1 \times I_{d-1} \times S_d \times I_{d+1} \times \cdots \times I_D}$ with elements



(a) Diagram representation of a scalar a, vector $\boldsymbol{a}$, matrix $\boldsymbol{A}$ and 3-way tensor $\boldsymbol{\mathcal{A}}$ **(b)** Diagram representation of equation (2) with all dimensions labelled

Fig. 1 Basic TN diagrams

$$\boldsymbol{\mathcal{B}}(i_1, \cdots, i_{d-1}, s_d, i_{d+1}, \cdots, i_D) := \sum_{i_d=1}^{I_d} \boldsymbol{\mathcal{A}}(i_1, \cdots, i_{d-1}, i_d, i_{d+1}, \cdots, i_D)\, \boldsymbol{U}_d(j_d, i_d).$$

A very convenient graphical representation of D-way tensors is shown in Fig. 1a. Tensors are here represented by nodes and each edge denotes a particular index of the tensor. The order of the tensor is then easily determined by counting the number of edges. Since a scalar is a zeroth-order tensor, it is represented by a node without any edges. The graphical representation of a summation over an index is by connecting the edge between the two nodes in the diagram. For example, the two index summations of a 3-way tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ with a matrix $\boldsymbol{U}_1 \in \mathbb{R}^{J_1 \times I_1}$ and a vector $\boldsymbol{u}_3 \in \mathbb{R}^{I_3}$

$$(\boldsymbol{\mathcal{A}} \times_1 \boldsymbol{U}_1 \times_3 \boldsymbol{u}_3^{\mathrm{T}}) = \sum_{i_1, i_3} \boldsymbol{\mathcal{A}}(i_1, :, i_3)\, \boldsymbol{U}_1(:, i_1)\, \boldsymbol{u}_3(i_3) \tag{2}$$

is graphically depicted in Fig. 1b by two connected edges between the nodes for $\boldsymbol{\mathcal{A}}, \boldsymbol{U}_1$ and $\boldsymbol{u}_3$. The result from these two summations is a $J_1 \times I_2$ matrix, which can also be deduced from the two "free" edges in Fig. 1b. Three important tensor decompositions in this article are the Tucker decomposition, the TT and the MERA. Each of these decompositions will now be briefly discussed.

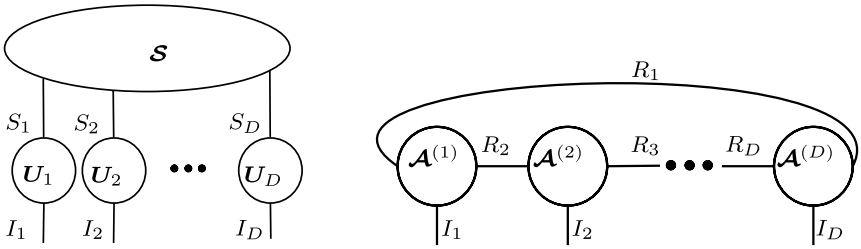
2.1 Tucker Decomposition

The Tucker decomposition represents a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_D}$ as

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{S}} \times_1 \boldsymbol{U}_1 \times_2 \cdots \times_D \boldsymbol{U}_D, \tag{3}$$

where $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{S_1 \times \cdots \times S_d}$ is called the Tucker core tensor and $\boldsymbol{U}_d \in \mathbb{R}^{I_d \times S_d}$ $(1 \leqslant d \leqslant D)$ are the Tucker factor matrices. The total storage complexity of the Tucker decomposition is therefore $\prod_{d=1}^{D} S_d + \sum_{d=1}^{D} I_d S_d$. These factor matrices are typically chosen to be orthogonal and can then be obtained as the left singular vectors of the corresponding unfolded matrices of $\boldsymbol{\mathcal{A}}$. A special case of the Tucker decomposition is the HOSVD [6], which has orthogonal matrices and where the Tucker core satisfies two additional properties. The dimensions $S_1, \cdots, S_D$ of the Tucker core are called the multilinear rank of $\boldsymbol{\mathcal{A}}$ and are defined as

$$S_d = \operatorname{rank}(\boldsymbol{A}_{<d>}) \leqslant I_d$$



(a) Diagram of a Tucker decomposition **(b)** Diagram of a TT

Fig. 2 Diagram representation of the Tucker and TT decompositions

for all values of d. A graphical representation of the Tucker decomposition is shown in Fig. 2a.

2.2 TT Decomposition

The TT decomposition was introduced into the scientific computing community in [21], but was known as a matrix product state in the field of condensed matter physics [23, 24] a decade earlier.

Definition 5 The TT decomposition of a given tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$ is a set of 3-way tensors $\mathcal{A}^{(d)} \in \mathbb{R}^{R_d \times I_d \times R_{d+1}}$ $(1 \leqslant d \leqslant D)$ with $R_1 = R_{D+1} = 1$ such that each entry $\mathcal{A}(i_1, i_2, \cdots, i_D)$ can be computed from

$$\sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_D=1}^{R_D} \mathcal{A}^{(1)}(r_1, i_1, r_2)\, \mathcal{A}^{(2)}(r_2, i_2, r_3) \cdots \mathcal{A}^{(D)}(r_D, i_D, r_1). \tag{4}$$

The 3-way tensors of the TT are also called the TT-cores and the minimal values of $R_1, \cdots, R_D$ for which (4) holds exactly for all tensor entries are called the TT-ranks. When $R_1 = R_{D+1} > 1$ the decomposition is called a tensor ring (TR), for which the diagram is shown in Fig. 2b with all dimensions of the TT-cores indicated. We will consider from now on only the TT case and, therefore, the R_1-link in Fig. 2b that "closes the loop" will not be drawn in future diagrams anymore. The total storage complexity of a TT is $\sum_{d=1}^{D} R_d R_{d+1} I_d$. The TT-ranks are upper bounded as described by the following theorem.

Theorem 1 (Theorem 2.1 of [22]) *For any tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_D}$ there exists a TT-decomposition with TT-ranks*

$$R_d \leqslant \min\left(\prod_{k=1}^{d-1} I_k, \prod_{k=d}^{D} I_k\right)$$

for $d = 2, \cdots, D-1$.

Suppose now that we have a Tucker core in the TT form. The mode-products of the Tucker factor matrices with this Tucker core in the TT form do not alter its TT-ranks. Theorem 1, therefore, reveals the connection between the upper bounds on the TT-ranks of a given tensor and its multilinear rank.

Corollary 1 *Let $\mathcal{A}$ be a D-way tensor with multilinear rank $S_1, \cdots, S_D$. Then, its TT-ranks $R_2, \cdots, R_D$ satisfy*

$$R_d \leqslant \min\left(\prod_{k=1}^{d-1} S_k, \prod_{k=d}^{D} S_k\right)$$

for $d = 2, \cdots, D-1$.

The TT approximation of a given tensor with a prescribed relative error can be computed with either the TT-SVD algorithm [21, p. 2301] or the TT-cross algorithm [22]. Furthermore, through the TT-rounding procedure [21, p. 2305] the TT-ranks of a given TT can be truncated such that the computed approximation satisfies a prescribed relative error. The notion of a TT in the site-d-mixed-canonical form will be very important in the development of the algorithms in this article and relies on both left-orthogonal and right-orthogonal TT-cores.

Definition 6 ([17, p. A689]) A TT-core $\mathcal{A}^{(d)}$ is left-orthogonal if it can be reshaped into an $R_d I_d \times R_{d+1}$ matrix $\boldsymbol{A}_d$ such that

$$\boldsymbol{A}_d^{\mathrm{T}} \boldsymbol{A}_d = \boldsymbol{I}_{R_{d+1}}.$$

Similarly, a TT-core $\mathcal{A}^{(d)}$ is right-orthogonal if it can be reshaped into an $R_d \times I_d R_{d+1}$ matrix $\tilde{\boldsymbol{A}}_d$ such that

$$\tilde{\boldsymbol{A}}_d \tilde{\boldsymbol{A}}_d^{\mathrm{T}} = \boldsymbol{I}_{R_d}.$$

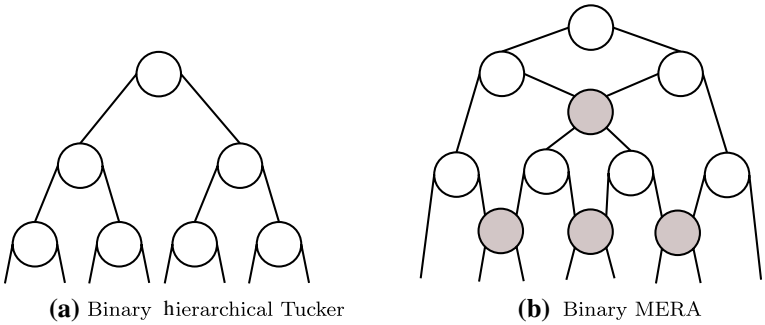
A TT is in the site-d-mixed-canonical form when all TT-cores $\mathcal{A}^{(1)}$ up to $\mathcal{A}^{(d-1)}$ are left-orthogonal and all TT-cores $\mathcal{A}^{(d+1)}$ up to $\mathcal{A}^{(D)}$ are right-orthogonal.

Once a TT is in the site-d-mixed-canonical form, then it can be readily verified that its Frobenius norm is easily obtained from the d-th core tensor

$$||\mathcal{A}||_{\mathrm{F}}^2 = ||\mathcal{A}^{(d)}||_{\mathrm{F}}^2.$$

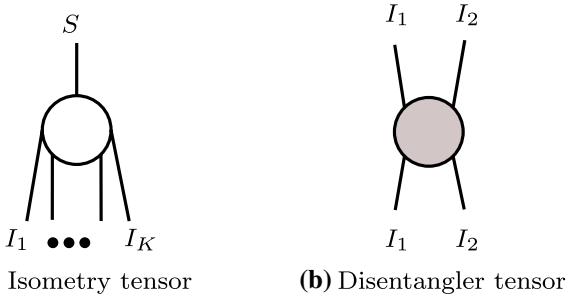
2.3 MERA

The MERA decomposition is a generalization of the hierarchical Tucker decomposition and consists of three different building blocks. A common implementation of the hierarchical Tucker decomposition is the binary tree form, as shown in Fig. 3a. Reading such a diagram from the bottom to the top, one can interpret each row/layer in such a tree structure as a coarse-graining transformation where each tensor in a row/layer transforms two indices into one index. Such tensors $\mathcal{W}$ of size $I_1 \times \cdots \times I_K \times S$ that reduce $K > 1$ indices



(a) Binary hierarchical Tucker **(b)** Binary MERA

Fig. 3 Diagram representation of two hierarchical tensor decompositions



(a) Isometry tensor **(b)** Disentangler tensor

Fig. 4 Diagram representation of two MERA building block tensors

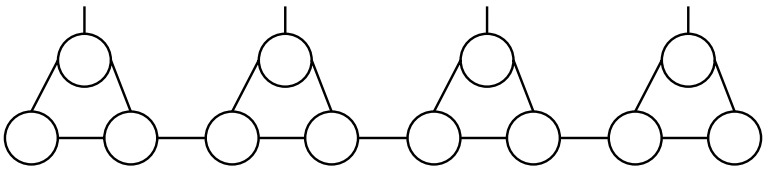


Fig. 5 A TT of an 8-way tensor (bottom row) is coarse-grained into a 4-way tensor through one layer of an HT/TTN

to a single index are called isometries. An isometry can always be reshaped into a size $I_1 I_2 \cdots I_K \times S$ matrix $\boldsymbol{W}$ with orthonormal columns

$$\boldsymbol{W}^{\mathrm{T}}\, \boldsymbol{W} = \boldsymbol{I}_S,$$

where S is the dimension of the "output" index. The minimal outgoing dimensions of all isometries such that the MERA represents a given tensor exactly are called the MERA-ranks. The diagram representation of an isometry is shown in Fig. 4a. The bottom layer of isometries with $K = 2$ in Fig. 3a reduces the eight indices of a given tensor into four indices, as illustrated in Fig. 5. Each application of a layer in the tree halves the resulting total number of indices. The coarse-graining with a hierarchical Tucker decomposition pairs two consecutive indices and sums over them, thereby ignoring possible correlations over neighbouring indices resulting in higher ranks during coarse-graining. This issue is resolved in the MERA through the introduction of additional disentangler tensors in the

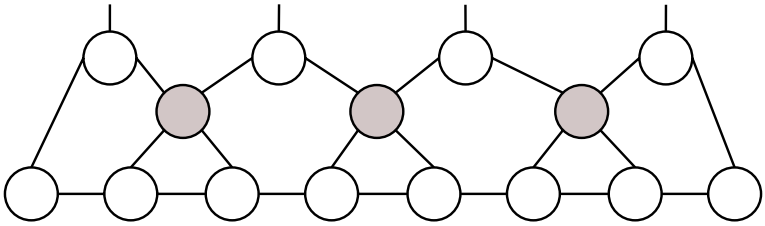


Fig. 6 A TT of an 8-way tensor (bottom row) is coarse-grained into a 4-way tensor through one layer of a MERA

coarse-graining layers. Disentanglers, shown as shaded nodes in Fig. 3b, "bridge" neighbouring pairs before being coarse-grained. A disentangler tensor is per definition a 4-way tensor $\mathcal{V}$ of size $I_1 \times I_2 \times I_1 \times I_2$ that can be reshaped into an orthogonal $I_1 I_2 \times I_1 I_2$ matrix V. The reduction of an 8-way TT into a 4-way TT through a MERA layer is shown as a diagram in Fig. 6. The third and final MERA building block is the top tensor. This tensor $\mathcal{T}$ is located at the top of the MERA structure and connects to all outgoing isometry indices of the highest layer. Since all disentanglers and isometries have their respective notion of orthogonality, it follows that the Frobenius norm of a tensor $\mathcal{A}$ that is represented by a MERA is given by $||\mathcal{A}||_{\mathrm{F}}^2 = ||\mathcal{T}||_{\mathrm{F}}^2$. This easy computation of the norm due to orthogonality is very similar to the case of a TT in the site-d-mixed canonical form. The storage complexity of a MERA is simply the sum of storage complexities of all disentanglers, isometries and the top tensor. In this respect, it is only meaningful from a data tensor compression perspective to have MERA-ranks that do not increase over consecutive layers. In the next section, we develop the main algorithm to convert a given TT into a Tucker decomposition and this algorithm will serve as the main computational building block to eventually convert a TT into a MERA.

3 TT to Tucker Decomposition

In this section, an algorithm is developed that converts a given TT into either an HOSVD or a truncated HOSVD with a guaranteed upper bound on the relative approximation error. The Tucker core $\mathcal{S}$ will be directly obtained in the TT format, avoiding its exponential storage complexity. The starting point of the algorithm is a TT in the site-1-mixed-canonical form. Before stating the algorithm, we first introduce some additional notation together with an important lemma.

3.1 Tucker Factor Matrix from TT-Core

To know how a given TT can be converted into a Tucker decomposition we need to know how the Tucker factor matrices can be computed from each TT-core. To describe this computation, we first introduce the following convenient notation.

Definition 7 Let $\mathcal{A}^{(1)}, \cdots, \mathcal{A}^{(D)}$ be TT-cores of a D-way tensor $\mathcal{A}$. We define $A_{<d}$ as the $R_d \times (I_1 \cdots I_{d-1})$ matrix obtained from summing over the auxiliary indices of

$\mathcal{A}^{(1)}$ up to $\mathcal{A}^{(d-1)}$ and permuting and reshaping the result into the desired matrix. The $R_{d+1} \times (I_{d+1} \cdots I_D)$ matrix $\boldsymbol{A}_{>d}$ is defined similarly from the TT-cores $\mathcal{A}^{(d+1)}$ up to $\mathcal{A}^{(D)}$. The $I_d \times (R_d R_{d+1})$ matrix $\boldsymbol{A}_d$ is defined from permuting and reshaping $\mathcal{A}^{(d)}$. Finally, both $\boldsymbol{A}_{<1}$ and $\boldsymbol{A}_{>D}$ are defined to be unit scalars.

Note that if the TT of $\mathcal{A}$ is in the site-d-mixed-canonical form, then the left- and right-orthogonality of the TT-cores implies that both $\boldsymbol{A}_{<d}$ and $\boldsymbol{A}_{>d}$ have orthonormal rows

$$\boldsymbol{A}_{<d}\, \boldsymbol{A}_{<d}^{\mathrm{T}} = \boldsymbol{I}_{R_d} \quad \text{and} \quad \boldsymbol{A}_{>d}\, \boldsymbol{A}_{>d}^{\mathrm{T}} = \boldsymbol{I}_{R_{d+1}}.$$

The following lemma tells us how the unfolding matrix $\boldsymbol{A}_{<d>}$ can be written in terms of the matrices from Definition 7.

Lemma 1 *For a D-way tensor $\mathcal{A}$ in the TT-form we have the following relationship*:

$$\boldsymbol{A}_{<d>} = \boldsymbol{A}_d \left(\boldsymbol{A}_{>d} \otimes \boldsymbol{A}_{<d}\right), \quad d = 1, \cdots, D.$$

The Kronecker product in Lemma 1 is due to the rank-1 link of the TT. Also note that the rows of $\left(\boldsymbol{A}_{>d} \otimes \boldsymbol{A}_{<d}\right)$ are orthonormal when the TT is in the site-d-mixed-canonical form, due to the preservation of orthonormality with the Kronecker product. Lemma 1 tells us that any unfolding matrix $\boldsymbol{A}_{<d>}$ can be written as a product of $\boldsymbol{A}_d$ with $\left(\boldsymbol{A}_{>d} \otimes \boldsymbol{A}_{<d}\right)$, which leads to the following two corollaries.

Corollary 2 *For a D-way tensor $\mathcal{A}$ with multilinear ranks $S_1, \cdots, S_D$ we have that*

$$S_d = \operatorname{rank}(\boldsymbol{A}_{<d>}) = \operatorname{rank}(\boldsymbol{A}_d) \leqslant \min(I_d, R_d R_{d+1}) \leqslant I_d, \quad d = 1, \cdots, D.$$

Corollary 3 *For a tensor $\mathcal{A}$ in the site-d-mixed-canonical TT-form, let the compact SVD of $\boldsymbol{A}_d$ be given by $\boldsymbol{U}_d\, \boldsymbol{S}\, \boldsymbol{V}^{\mathrm{T}}$. Then, the compact SVD of the unfolding matrix $\boldsymbol{A}_{<d>}$ is*

$$\boldsymbol{A}_{<d>} = \boldsymbol{U}_d\, \boldsymbol{S}\, \boldsymbol{V}^{\mathrm{T}} \left(\boldsymbol{A}_{>d} \otimes \boldsymbol{A}_{<d}\right).$$

In Corollary 2 we have tacitly assumed that $R_d R_{d+1} < \prod_{k \neq d} I_k$ is always satisfied. Corollary 3 follows directly from the fact that the product of matrices with orthonormal rows also has orthonormal rows. The matrix $\left(\boldsymbol{A}_{>d}^{\mathrm{T}} \otimes \boldsymbol{A}_{<d}^{\mathrm{T}}\right)\boldsymbol{V}$ therefore contains the right singular vectors of $\boldsymbol{A}_{<d>}$ corresponding with the S_d largest singular values. Corollary 3 also implies that the HOSVD factor matrix $\boldsymbol{U}_d$ can be directly computed from the SVD of $\boldsymbol{A}_d$. The d-th component S_d of the multilinear rank can be determined by inspecting the singular values on the diagonal $\boldsymbol{S}$ matrix. If there is no need to know the exact multilinear rank, then a square orthogonal $\boldsymbol{U}_d$ can also be obtained through a QR decomposition of $\boldsymbol{A}_d$.

3.2 The TT to Tucker Conversion Algorithm

Lemma 1 forms the basis of the proposed algorithm to convert a given TT into either an HOSVD or a truncated HOSVD. The algorithm to compute a truncated HOSVD is presented in pseudo-code as Algorithm 1. The algorithm assumes the TT is in the site-1-mixed-canonical form but can be easily adjusted to work for any other starting site. The main idea of Algorithm 1 is to compute the orthogonal factor matrix $\boldsymbol{U}_d$ using Corollary 3 and then to bring the TT into the site-$(d+1)$-mixed-canonical form. The conversion of the

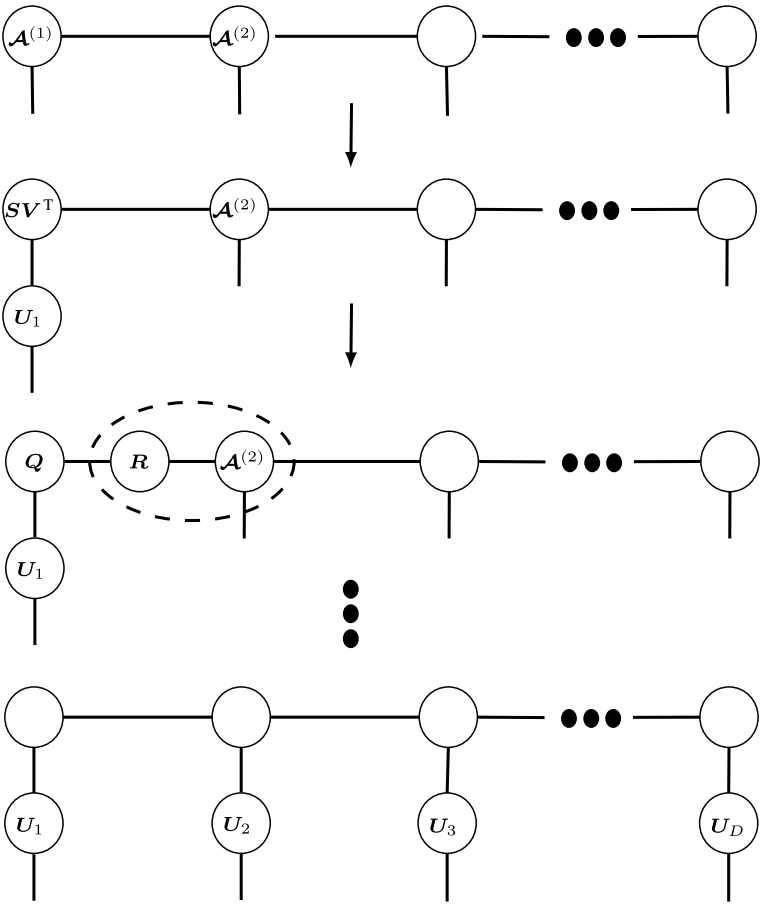


Fig. 7 The complete first execution of the for-loop in Algorithm 1 in diagram form

TT into the site-$(d+1)$-)mixed-canonical form is computed through a QR decomposition of the $\boldsymbol{SV}^{\mathrm{T}}$ factor. The orthogonal $\boldsymbol{Q}$ matrix is then retained as the d-th TT-core of the Tucker core $\boldsymbol{\mathcal{S}}$, while the norm of $\boldsymbol{\mathcal{A}}$ is moved to the next TT-core $\boldsymbol{\mathcal{A}}^{(d+1)}$ through the absorption of the $\boldsymbol{R}$ factor. The final TT of $\boldsymbol{\mathcal{S}}$ will, therefore, be in the site-D-mixed-canonical form. Both the SVD step and the QR decomposition step are graphically represented in Fig. 7. During each run of the for-loop in Algorithm 1 we are working with a partially truncated core tensor, which is very reminiscent of the ST-HOSVD algorithm [28]. In fact, the approximation error induced by truncating the SVD in Algorithm 1 can also be expressed exactly in terms of the singular values.

Theorem 2 *Let $\sigma_d(i_d)$ be the i_d-th singular value of $\mathbf{A}_d$ and $\hat{\boldsymbol{\mathcal{A}}}$ be the tensor computed by Algorithm* 1 *with truncated SVDs. Then,*

$$||\boldsymbol{\mathcal{A}} - \hat{\boldsymbol{\mathcal{A}}}||_{\mathrm{F}}^2 = \sum_{d=1}^{D} \sum_{i_d=S_d+1}^{I_d} \sigma_d(i_d)^2.$$

Given that the TT for the all-orthogonal Tucker core is in the site-d-mixed-canonical form and the similarity of Algorithm 1 concerning the use of a sequentially truncated Tucker core, it follows that the proof of Theorem 2 is completely identical with the one found in [28, p. A1039]. Theorem 2 allows us to compute the absolute approximation error for a truncated HOSVD during the execution of Algorithm 1 by simply adding the squares of the discarded singular values. In addition, Theorem 2 also allows us to compute a truncated HOSVD for a given upper bound ϵ on the relative approximation error. Since Algorithm 1 consists of D truncated SVDs, setting the tolerance δ for each of these SVDs to $\epsilon||\boldsymbol{\mathcal{A}}||_{\mathrm{F}}/\sqrt{D}$ then effectively guarantees that the computed approximation $\boldsymbol{\mathcal{B}}$ satisfies $||\boldsymbol{\mathcal{A}}-\boldsymbol{\mathcal{B}}||_{\mathrm{F}} \leqslant \epsilon||\boldsymbol{\mathcal{A}}||_{\mathrm{F}}$. Alternatively, if one is interested in the computation of an HOSVD without any truncation, then it suffices to replace the SVD in line 3) of Algorithm 1 with a QR decomposition.

Algorithm 1 Convert TT into truncated HOSVD

Input TT $\boldsymbol{\mathcal{A}}^{(1)}, \cdots, \boldsymbol{\mathcal{A}}^{(D)}$ in the site-1-mixed-canonical form of tensor $\boldsymbol{\mathcal{A}}$, accuracy ϵ.
Output Tucker core $\boldsymbol{\mathcal{S}}$ in the site-D-mixed-canonical form, orthogonal factor matrices $\boldsymbol{U}_1, \cdots, \boldsymbol{U}_D$ of approximation $\boldsymbol{\mathcal{B}}$ such that $||\boldsymbol{\mathcal{A}}-\boldsymbol{\mathcal{B}}||_{\mathrm{F}} \leqslant \epsilon||\boldsymbol{\mathcal{A}}||_{\mathrm{F}}$.

1) $\delta \leftarrow \frac{\epsilon\,||\boldsymbol{\mathcal{A}}||_{\mathrm{F}}}{\sqrt{D}}$
2) **for** $d = 1 : D$ **do**
3) $\quad \boldsymbol{U}_d, \boldsymbol{S}, \boldsymbol{V}^{\mathrm{T}} \leftarrow \mathrm{SVD}_\delta(\boldsymbol{A}_d)$ % $\boldsymbol{A}_d = \boldsymbol{U}_d\,\boldsymbol{S}\,\boldsymbol{V}^{\mathrm{T}} + \boldsymbol{E}$, $||\boldsymbol{E}||_{\mathrm{F}} \leqslant \delta$, $S_d = \mathrm{rank}(\boldsymbol{S})$.
4) $\quad \boldsymbol{\mathcal{T}} \leftarrow \mathrm{reshape}(\boldsymbol{S}\boldsymbol{V}^{\mathrm{T}}, [S_d, R_d, R_{d+1}])$
5) $\quad \boldsymbol{\mathcal{T}} \leftarrow \mathrm{permute}(\boldsymbol{\mathcal{T}}, [2, 1, 3])$
6) $\quad$ **if** $d == D$ **then**
7) $\quad\quad \boldsymbol{\mathcal{S}}^{(d)} \leftarrow \boldsymbol{\mathcal{T}}$
8) $\quad$ **else**
9) $\quad\quad \boldsymbol{T} \leftarrow \mathrm{reshape}(\boldsymbol{\mathcal{T}}, [R_d S_d, R_{d+1}])$
10) $\quad\quad \boldsymbol{Q}, \boldsymbol{R} \leftarrow \mathrm{QR}(\boldsymbol{T})$
11) $\quad\quad \boldsymbol{\mathcal{S}}^{(d)} \leftarrow \mathrm{reshape}(\boldsymbol{Q}, [R_d, S_d, R_{d+1}])$
12) $\quad\quad \boldsymbol{\mathcal{A}}^{(d+1)} \leftarrow \boldsymbol{\mathcal{A}}^{(d+1)} \times_1 \boldsymbol{R}$
13) $\quad$ **end if**
14) **end for**

3.3 Computational Complexity

In this subsection we briefly analyze the computational complexity of Algorithm 1. For notational convenience, we will assume that a D-way tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I\times\cdots\times I}$ is represented by a TT with uniform TT-rank R. An additional assumption is that $I < R^2$. The computation of D thin SVDs of $\boldsymbol{A}_d \in \mathbb{R}^{I\times R^2}$ in line 3) takes then $D(14R^2I^2 + 8I^3)$ flops [11, p.493]. The QR decompositions in line 10) required for the computation of the site-$(d+1)$-mixed-canonical form require $(D-1)\big(2I^2(R^2 - I/3) + 4(R^4I - R^2I^2 + I^3/3)\big)$ flops [11, p.249] when performed with Householder transformations. In practical cases, we have that $I \leqslant R^2$ and this implies that the total computational complexity for Algorithm 1 is dominated by the $O(R^4I)$ term of the QR decompositions. If instead of a guaranteed relative approximation error a Tucker decomposition with given multilinear-rank is desired, then one can replace the SVD in line 3) of Algorithm 1 by a randomized SVD [14] or an implicitly restarted Arnoldi method [20]. Also note that the actual complexity will depend heavily on the order of the indices, which is also the case with the sequentially truncated HOSVD. In practice, a heuristic that reduces the computational complexity is to permute the dimensions of the tensor $\boldsymbol{\mathcal{A}}$ in an ascending manner prior

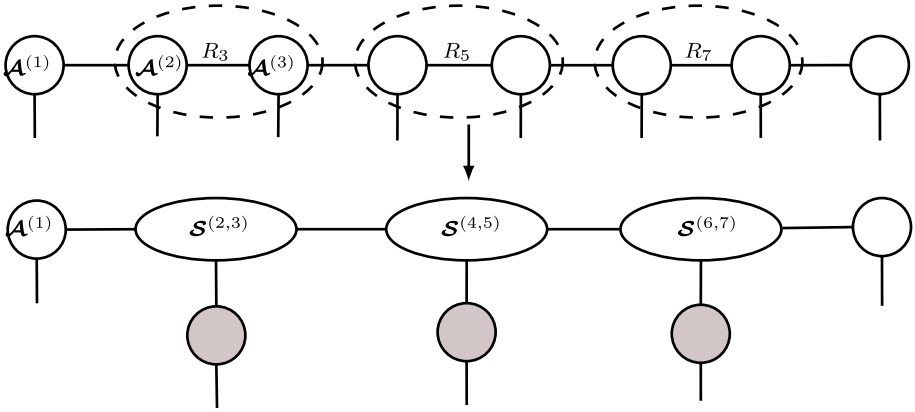


Fig. 8 Diagram of disentangler computation through an HOSVD step in the TT format as described in Algorithm 2

to computing its Tucker decomposition [28, p. A1041] as this permutation typically reduces the maximal value of R.

A Tucker decomposition where the Tucker core tensor is stored as a TT was first introduced in [7]. Algorithm 5 [7, p. 611] describes how such a decomposition can be obtained by means of an iterative ALS method. One disadvantage of an ALS approach, however, is that the desired TT-ranks need to be chosen a priori. An alternative DMRG approach that is able to retrieve the TT-ranks has been proposed but this comes at the cost of a computational complexity of $O(R^3 I^3)$ [7, p. 612].

4 TT to MERA

The conversion of a TT into a MERA can be done via a sequence of HOSVD and truncated HOSVD computations. The disentanglers are computed through an HOSVD while the isometries are obtained through a truncated HOSVD. The conversion algorithm will be demonstrated through an illustrative example that consists of a TT with eight TT-cores with dimensions $R_d \times I \times R_{d+1}$ for $d = 1, \cdots, 8$. The goal is to compute a MERA for which the isometries convert $K = 2$ indices into one. As demonstrated in Fig. 6, the "action" of the first MERA layer is the application of three disentanglers. The diagram representation of the required operations to find these disentanglers is shown in Fig. 8. The required disentanglers are orthogonal transformations on three index pairs. The relevant TT-cores are contracted over their auxiliary indices R_3, R_5, R_7 to obtain so-called "supercores". For example, TT-cores $\mathcal{A}^{(2)}$ and $\mathcal{A}^{(3)}$ are combined into a supercore $\mathcal{A}^{(2,3)} \in \mathbb{R}^{R_2 \times I^2 \times R_4}$, where the two free indices of size I are combined into one multi-index of size I^2. Algorithm 1 is then applied to these supercores with a full SVD to obtain the desired disentanglers. The bottom row of Fig. 8 shows the obtained partial Tucker core with the orthogonal factor matrices, which will serve as the transposes of the disentanglers. For example, Algorithm 1 allows us to write

$$\mathcal{A}^{(2,3)} = \mathcal{S}^{(2,3)} \times_2 \boldsymbol{U}_{2,3},$$

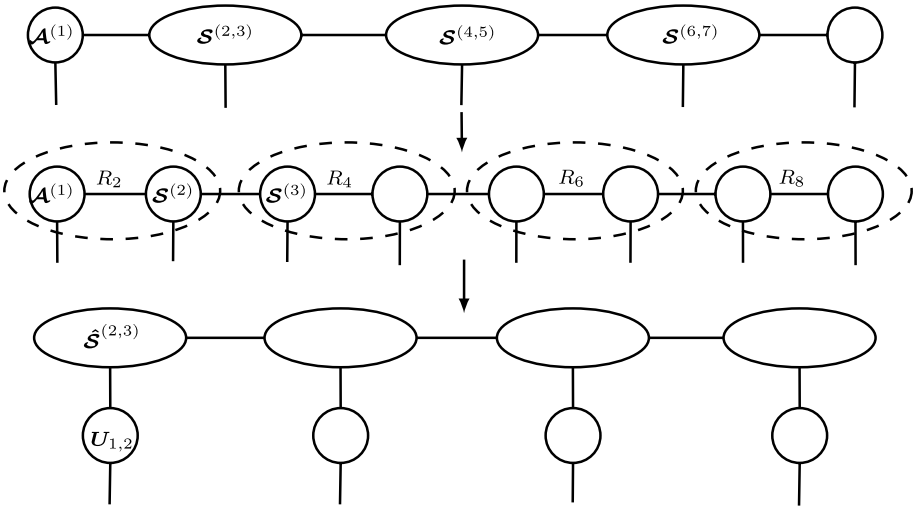


Fig. 9 Diagram of isometry computation through a truncated HOSVD step in the TT format as described in Algorithm 2

where $\mathcal{S}^{(2,3)} \in \mathbb{R}^{R_2 \times I^2 \times R_4}$ is represented by the leftmost oval of the bottom row in Fig. 8 and $\boldsymbol{U}_{2,3} \in \mathbb{R}^{I^2 \times I^2}$ is an orthogonal matrix. The desired disentangler is then obtained by reshaping $\boldsymbol{U}_{2,3}^{\mathrm{T}}$ into a cubical 4-way tensor of dimension I. The partial Tucker core is now used as the starting point for obtaining the isometries, as shown in the top row of Fig. 9. The supercores, represented by the ovals in the top row of Fig. 9, first need to be split back into separate TT-cores through an SVD, e.g., the supercore $\mathcal{S}^{(2,3)}$ is reshaped into the $R_2 I \times I R_4$ matrix $\boldsymbol{S}_{2,3}$,

$$\begin{aligned} \boldsymbol{S}_{2,3} &= \boldsymbol{U}\, \boldsymbol{S}\, \boldsymbol{V}^{\mathrm{T}} \\ &= \boldsymbol{S}_2\, \boldsymbol{S}_3 \end{aligned}$$

with $\boldsymbol{S}_2 := \boldsymbol{U}$ and $\boldsymbol{S}_3 := \boldsymbol{S}\boldsymbol{V}^{\mathrm{T}}$. The rank R_3 is determined as the number of nonzero singular values such that $\boldsymbol{S}_2 \in \mathbb{R}^{R_2 I \times R_3}$ and $\boldsymbol{S}_3 \in \mathbb{R}^{R_3 \times I R_4}$. The desired TT-cores are obtained by reshaping $\boldsymbol{S}_2$, $\boldsymbol{S}_3$ into the desired 3-way tensors. In this way we arrive at the second row from the top of Fig. 9. In a MERA with $K = 2$, there are the isometries orthogonal transformations that convert two consecutive TT indices into one index. The next step is therefore to form new supercores by summing over auxiliary indices R_2, R_4, R_6 and R_8. Applying Algorithm 1 with a truncated SVD then results in the desired isometries. Indeed, the first supercore $\hat{\mathcal{A}}^{(1,2)}$ can then be written as

$$\hat{\mathcal{A}}^{(1,2)} = \hat{\mathcal{A}}^{(1,2)} \times_2 \boldsymbol{U}_{1,2}$$

with $\hat{\mathcal{A}}^{(1,2)} \in \mathbb{R}^{1 \times S^2 \times R_3}$ and $\boldsymbol{U}_{1,2} \in \mathbb{R}^{I^2 \times S}$. The bottom row of Fig. 9 shows the diagram of the truncated HOSVD in the TT form. The desired isometry is obtained by reshaping $\boldsymbol{U}_{1,2}$ into an $I \times I \times S$ matrix, where S is the truncated index. Theorem 2 allows us to quantify

the absolute approximation error due to truncation at each isometry step in the formation of the MERA and compute a MERA that approximates a given tensor with a guaranteed relative error. If sufficient MERA layers have been computed through this procedure, then the remaining Tucker core can be retained as the top tensor. This final step also ensures that the norm of the MERA is completely determined by the top tensor. The pseudocode for the whole algorithm is presented in Algorithm 2.

Algorithm 2 MERACLE: convert TT into a MERA

Input TT $\boldsymbol{\mathcal{A}}^{(1)}, \cdots, \boldsymbol{\mathcal{A}}^{(D)}$ in the site-K-mixed-canonical form, order K of the isometries, accuracy ϵ.
Output Isometries, disentanglers and top tensor of a MERA $\boldsymbol{\mathcal{B}}$ such that $||\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{B}}||_{\mathrm{F}} \leqslant \epsilon ||\boldsymbol{\mathcal{A}}||_{\mathrm{F}}$.

1) $N \leftarrow$ total number of isometries in the MERA.
2) $\delta \leftarrow \frac{\epsilon ||\boldsymbol{\mathcal{A}}||_{\mathrm{F}}}{\sqrt{N}}$.
3) **for** each MERA layer **do**
4) Compute supercores of the TT according to desired disentangler locations.
5) Apply Algorithm 1 with full SVD on all supercores.
6) Retrieve disentanglers from the orthogonal HOSVD factor matrices.
7) Split supercores of the partial Tucker core with the SVD.
8) Compute new supercores according to the location and order K of the isometries.
9) Apply Algorithm 1 with a truncated SVD_δ on all supercores.
10) Retrieve isometries from the truncated HOSVD factor matrices.
11) **if** final MERA layer **then**
12) Retain single Tucker core tensor as the top tensor.
13) **else**
14) Split supercores of the Tucker core with an SVD.
15) **end if**
16) **end for**

5 Iterative Algorithm for Finding a Rank-Lowering Disentangler

Assuming that an exact low-rank MERA exists for a given TT, Algorithm 2 will typically fail to find it. In practice, the output dimensions S of the isometries will simply be the product of the input dimensions $I_1 I_2 \cdots I_K$ and no truncation is ever performed. This leads to an exponential growth of the isometry output dimensions as a function of the number of MERA layers. The problem with Algorithm 2 is that it fails to find the correct disentanglers. To explain the issue at hand, we first need to explain the workings of a disentangler in a bit more detail.

5.1 Disentangler

As mentioned earlier in Sect. 2.3, disentanglers were originally introduced to remove possible correlations between neighbouring indices to avoid high TT-ranks after coarse-graining [30]. Figure 10 illustrates the key effect of a disentangler on a simple example of four TT-cores with dimensions $I_1 = I_2 = I_3 = I_4 = I$. Note that the maximal TT-rank R between the second and third TT-cores is I^2. Suppose that $R = I^2$. It is straightforward to see that having no disentangler implies that the output dimensions of the two isometries needs to

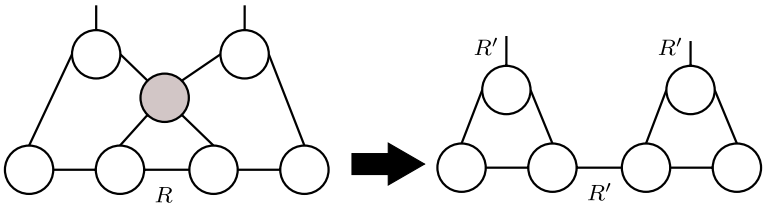


Fig. 10 The disentangler reduces the TT-rank from R to R' with $R < R'$, allowing the two isometries to truncate to R' without any loss of accuracy

be $R = I^2$, as two indices with dimension I are simply combined into one multi-index. Now suppose that prior to the isometries, a disentangler can be applied to the second and third TT-cores such that the TT-rank R is reduced to $R' < R$. In this case, the two isometries can truncate the dimensions I^2 down to R' without the loss of any accuracy. Unfortunately, the disentanglers obtained from an HOSVD in Algorithm 2 do not reduce the TT-ranks, which implies that none of the isometries can effectively truncate the dimensions. If we are able to develop an algorithm that can find a rank-lowering disentangler, then Corollary 2 automatically guarantees that the truncated HOSVD in line 9) of Algorithm 2 will find an optimal isometry. In the next subsection, we propose an iterative algorithm that attempts to recover rank-lowering disentanglers.

5.2 Iterative Orthogonal Procrustes Algorithm

Before stating the problem of finding the optimal disentangler in a formal way, we first introduce some convenient notation.

Definition 8 For a supercore $\mathcal{A}^{(d,d+1)} \in \mathbb{R}^{R_d \times I_d I_{d+1} \times R_{d+2}}$ we define the following matricizations:

$$\begin{aligned} \boldsymbol{A}^{(d,d+1)} &\in \mathbb{R}^{R_d I_d \times I_{d+1} R_{d+2}}, \\ \boldsymbol{A} &\in \mathbb{R}^{I_d I_{d+1} \times R_d R_{d+2}}. \end{aligned}$$

These matrices are per definition related to one another via the shuffling operator shuf and its inverse

$$\begin{aligned} \boldsymbol{A} &= \mathrm{shuf}\big(\boldsymbol{A}^{(d,d+1)}\big), \\ \boldsymbol{A}^{(d,d+1)} &= \mathrm{shuf}^{-1}(\boldsymbol{A}). \end{aligned}$$

With these definitions the optimal disentangler problem can now be formulated.

Problem 1 Given a supercore $\mathcal{A}^{(d,d+1)}$ for which $\mathrm{rank}\big(\boldsymbol{A}^{(d,d+1)}\big) = R$, find an orthogonal matrix $\boldsymbol{V} \in \mathbb{R}^{I_d I_{d+1} \times I_d I_{d+1}}$ such that

$$\boldsymbol{A}' := \boldsymbol{V}\,\mathrm{shuf}\big(\boldsymbol{A}^{(d,d+1)}\big) = \boldsymbol{V}\boldsymbol{A}$$

with $\mathrm{rank}\big(\mathrm{shuf}^{-1}\big(\boldsymbol{A}'\big)\big) = R' < R$.

Problem 1 is essentially an orthogonal Procrustes problem in $\boldsymbol{A}$ with the additional constraint that the orthogonal transformation $\boldsymbol{V}$ lowers the rank of $\boldsymbol{A}^{(d,d+1)}$. The difficulty is that both $\boldsymbol{A}'$ and $\text{shuf}^{-1}\left(\boldsymbol{A}'\right)$ are unknown. We therefore propose to solve the orthogonal Procrustes problem in an iterative manner, where we fix $\boldsymbol{A}^{(k,k+1)'}$ in every iteration to a low-rank approximation of $\boldsymbol{A}^{(d,d+1)}$. The computational complexity of solving the orthogonal Procrustes problem every iteration is $O((I_d I_{d+1})^3)$, as this amounts to computing the SVD of $\boldsymbol{A}'\boldsymbol{A}^{\mathrm{T}}$. The proposed iterative algorithm is presented in pseudocode as Algorithm 3. The stopping criterion can be set to a fixed maximum number of iterations or one can inspect the rank-gap $\sigma_{R'}/\sigma_{R'+1}$ of $\boldsymbol{A}^{(d,d+1)}$ and stop the iterations as soon as this gap has reached a certain order of magnitude. A low-rank approximation of $\boldsymbol{A}^{(d,d+1)}$ can be computed via its SVD. At this moment, there is no formal proof of convergence for Algorithm 3, nor is it known what the conditions for convergence are. The best that we are currently able to do is to empirically show the successful application of this algorithm and to explore its properties based on extensive numerical experiments.

Algorithm 3 Iterative disentangler computation

Input supercore $\boldsymbol{\mathcal{A}}^{(d,d+1)} \in \mathbb{R}^{R_d \times I_d I_{d+1} \times R_{d+2}}$.
Output Disentangler $\boldsymbol{\mathcal{V}}$ that reduces the TT-rank to R'.

1) $\boldsymbol{V} \leftarrow \boldsymbol{I}$ % initialize with identity matrix
while stopping criterion not true **do**
2) $\boldsymbol{A}^{(d,d+1)'} \leftarrow$ low-rank approximation of $\boldsymbol{A}^{(d,d+1)}$.
3) $\boldsymbol{A} \leftarrow \text{shuf}\left(\boldsymbol{A}^{(d,d+1)}\right)$ and $\boldsymbol{A}' \leftarrow \text{shuf}\left(\boldsymbol{A}^{(d,d+1)'}\right)$.
4) $\hat{\boldsymbol{V}} \leftarrow$ solve orthogonal Procrustes problem that minimizes $||\boldsymbol{V}\boldsymbol{A} - \boldsymbol{A}'||_{\mathrm{F}}^2$.
5) $\boldsymbol{A} \leftarrow \hat{\boldsymbol{V}}\,\boldsymbol{A}$.
6) $\boldsymbol{A}^{(d,d+1)} \leftarrow \text{shuf}^{-1}(\boldsymbol{A})$.
7) $\boldsymbol{V} \leftarrow \hat{\boldsymbol{V}}\,\boldsymbol{V}$.
end while
8) $\boldsymbol{\mathcal{V}} \leftarrow \text{reshape}(\boldsymbol{V}, [I_d, I_{d+1}, I_d, I_{d+1}])$.

6 Experiments

In this section, we demonstrate the computational efficiency of Algorithms 1, 2 and 3 through numerical experiments. All algorithms were implemented in MATLAB and the experiments were performed on a desktop computer with a 4-core processor running at 3.6 GHz with 16 GB RAM. The data files required to reproduce these experiments are available at [1] and an open source Matlab implementation of the algorithms is available at https://github.com/kbatseli/MERACLE.

6.1 Converting a TT into Tucker—Compression of Simulation Results

In this experiment, we demonstrate Algorithm 1 and how a representation of a Tucker decomposition can benefit compression without loss of accuracy. Inspired by the example discussed in [28, p. A1047], a tensor decomposition is used for the compression of the solution $u(x, y, t)$ of

Table 1 Comparison of different tensor decompositions in compressing the results from a numerical simulation of the 2D heat equation

	Time/s	Relative error	Compression
STHOSVD (3-way)	3.279	4.74 E−4	1 055
TT (3-way)	3.579	8.54 E−4	1 101
TT-Tucker (3-way)	0.019	8.54 E−4	1 116
STHOSVD (16-way)	26.710	6.68 E−4	3.255
TT (16-way)	8.325	6.11 E−4	12 572
TT-Tucker (16-way)	0.005	6.11 E−4	12 229

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

on the unit square $[0, 1]^2$ with the boundary condition $0.25 - |0.5 - x| \cdot |0.5 - y|$, which also describes the initial temperature distribution over the entire square. The PDE was discretized with a uniform mesh with the cell size $(\Delta s, \Delta s, \Delta t)$ and solved with the explicit Euler method, using a time step $0.25\Delta s^2$ to ensure numerical stability. We set $\Delta s = 10^{-2}$ and $\Delta t = 0.25 \cdot 10^{-4}$ and simulate for about 0.25 seconds, resulting in a tensor of size $100 \times 100 \times 10\,000$. The upper bound on the relative approximation error when computing tensor decompositions is set to 10^{-3}. We compare the sequentially truncated HOSVD (STHOSVD) with both the TT and Tucker decomposition in the TT form. The STHOSVD is computed with the MLSVD command of the Tensorlab toolbox [29], while the conversion of the original data tensor into a TT is done via the TT-SVD algorithm [21, p. 2301]. The TT is converted into a Tucker decomposition via Algorithm 1. We consider two cases. In the first case, we compute the three tensor decompositions on the original solution tensor, while in the second case we first reshape the original data into a 16-way tensor by factorization of all dimensions into their prime components. All results are shown in Table 1. The compression column contains the ratio between how many numbers are required to store the original tensor and how many numbers are required to store the decomposition. Not much difference in neither the total runtime, relative error or compression can be observed when the simulation solution is kept as a 3-way tensor. The computation of an STHOSVD of the 16-way tensor takes about 3 times longer than computing the corresponding TT. The resulting decomposition is also not able to compress the data very much as each of the dimensions of the 16-way tensor consist of (small) prime factors. The TT and Tucker decomposition in the TT form, however, result in a saving of around 12 000, which is an improvement of more than 10 times compared to the 3-way case. The time required for Algorithm 1 to compute the Tucker decomposition was in both cases negligible compared to the runtime of the TT-SVD algorithm.

6.2 Comparison of HOSVD with Algorithm 3

In this experiment, we compare Algorithm 1 with Algorithm 3 to retrieve a rank-lowering disentangler. For this we consider the MERA consisting of a single layer as depicted in Fig. 11. The top tensor is taken to be an $R' \times R'$ grayscale image[1]. In this particular case,

[1] The image was taken from http://absfreepic.com/free-photos/download/landscape-with-lake-4412x2941_12692.html, cropped and scaled to appropriate dimensions.

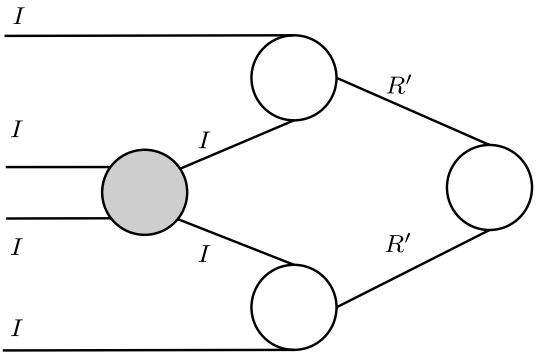
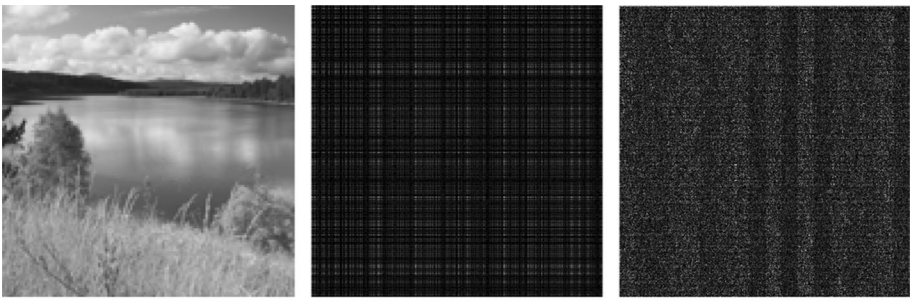


Fig. 11 The disentangler reduces the TT-rank from R to R' with $R > R'$, allowing the two isometries to truncate to R' without any loss of accuracy



(a) Top tensor image **(b)** Image after isometries **(c)** Image after disentangler

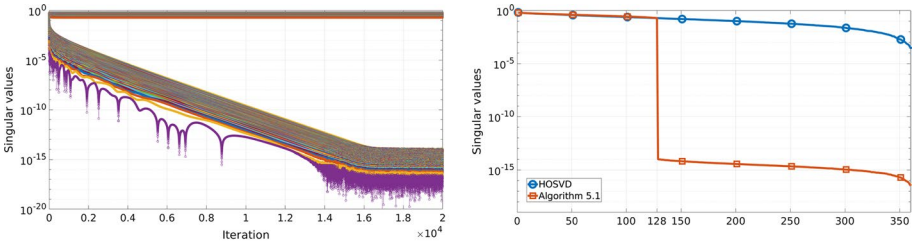
Fig. 12 Image after consecutive application of isometries and a disentangler

we set $R' = 128$. Both the $I \times I \times R'$ isometry tensors $\mathcal{W}$ and $I \times I \times I \times I$ disentangler tensor $\mathcal{V}$ are found from the orthogonalization of random matrices with appropriate sizes. For this experiment, we set $I = 19$ and choose the isometries to be identical. Given the $R' \times R'$ grayscale image top tensor $\boldsymbol{A}$ shown in Fig. 12a, we can now apply the MERA "backwards". The application of the two isometries on $\boldsymbol{A}$ is

$$\boldsymbol{B} = \boldsymbol{W}\boldsymbol{A}\,\boldsymbol{W}^{\mathrm{T}},$$

resulting in an $I^2 \times I^2$ image $\boldsymbol{B}$, shown in Fig. 12b. The corresponding TT of the image $\boldsymbol{B}$ has TT-ranks 19 and $128 = R'$. The application of the disentangler is then performed from the following steps:

$$\begin{aligned}
\mathcal{B} &:= \text{reshape}(\boldsymbol{B}, [I, I, I, I]),\\
\mathcal{B}_p &:= \text{permute}(\mathcal{B}, [2, 3, 4, 1]),\\
\tilde{\boldsymbol{B}} &:= \text{reshape}(\mathcal{B}_p, [I^2, I^2]),\\
\boldsymbol{C}_p &:= \boldsymbol{V}\tilde{\boldsymbol{B}},\\
\mathcal{C}_p &:= \text{reshape}(\boldsymbol{C}_p, [I, I, I, I]),\\
\mathcal{C} &:= \text{permute}(\mathcal{C}_p, [4, 1, 2, 3]),\\
\boldsymbol{C} &:= \text{reshape}(\mathcal{C}, [I^2, I^2]),
\end{aligned}$$



(a) Convergence of singular values during Algorithm 5.1

(b) Singular value decay of $\boldsymbol{C}^{(2,3)}$ after application of disentanglers

Fig. 13 Singular value graphs

Table 2 Minimal value of R' for which Algorithm 3 converges as a function of I

I	2	3	4	5	6	7	8	9	10	11	12	13	14
R'_{min}	2	4	6	9	12	16	20	25	30	37	44	51	59

resulting in the $I^2 \times I^2$ image $\boldsymbol{C}$ shown in Fig. 12c. The corresponding TT of the image $\boldsymbol{C}$ has TT-ranks 19 and $361 = I^2$. This increase of the TT-rank is reflected in the image as being much more "noisy" while the low-rank image of Fig. 12b has a particular block structure pattern. We now compare the use of Algorithm 1 with Algorithm 3 for retrieving a disentangler that is able to reduce the maximal TT-rank from 361 down to 128. Algorithm 3 is run on the TT of $\boldsymbol{\mathcal{C}}$ in site-4-mixed-canonical form and a rank-128 approximation of $\boldsymbol{C}^{(2,3)}$ is used. Each iteration of Algorithm 3 took 0.03 seconds and, as shown in Fig. 13a, about 16 000 iterations were required for the 233 smallest singular values to converge to values of about 10^{-15}. The computed disentanglers are then applied to the supercore $\boldsymbol{\mathcal{C}}^{(2,3)}$. The singular value decay of each corresponding $\boldsymbol{C}^{(2,3)}$ is shown in Fig. 13b, where it can be clearly seen that Algorithm 3 is able to retrieve a disentangler that lowers the rank to the minimal value of 128.

6.3 Limitations of Algorithm 3

We revisit the example from Sect. 6.2 and explore the validity of Algorithm 3 for different values of R' and I, as it is yet unclear under which conditions we are able to retrieve an exact rank-lowering disentangler. If I is fixed, then the rank of $\boldsymbol{C}$ is I^2 for the particular MERA of Sect. 6.2 and it appears that there exists a minimal value R'_{min} such that Algorithm 3 does not converge for values $R' < R'_{\text{min}}$. There is, however, an exception to this observation in that Algorithm 3 always converges if $R' = 1$. Table 2 lists all values of R'_{min} for values of I going from 2 up to 14, where the convergence of Algorithm 3 was determined from inspecting the singular value decay as in Fig. 13a. A first observation is that R'_{min} grows slowly compared to $R = I^2$, which implies that the range of values of R' for which Algorithm 3 converges gets larger as I grows. The reason for the existence of this R'_{min} is yet to be fully understood.

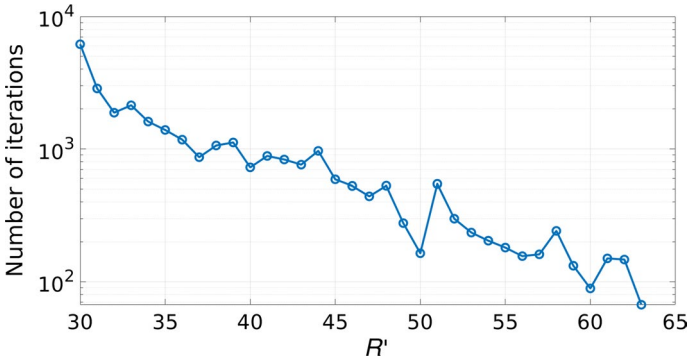


Fig. 14 The number of iterations required for Algorithm 3 to reach a rank-gap of $\sigma_{R'}/\sigma_{R'+1} = 10^{12}$ as a function of R' when $I = 8$

Table 3 Comparison of storage requirement and compression capability between a TT and a MERA for a 12th-order cubical tensor

	Storage requirement	Compression
Original tensor	10^{12}	1
TT	15 620 200	6.40×10^4
MERA	54 750	1.82×10^7

A second observation relates to the rate of convergence. It turns out that Algorithm 3 converges faster as the difference between R and R' becomes smaller. This is illustrated in Fig. 14 where the number of iterations required for Algorithm 3 to reach a rank-gap of $\sigma_{R'}/\sigma_{R'+1} = 10^{12}$ is shown for varying R' when $I = 8$. An approximately exponential growth in the number of required iterations can be seen as the difference of R' with $R = 8^2 = 64$ grows larger. This exponential growth might explain the existence of R'_{min} as a value of R' for which convergence becomes "infinitely slow". These observations will serve as a starting point to investigate the exact nature of why and when Algorithm 3 works, apart from the empirical study herein.

6.4 Comparison of Compression Capability Between a TT and a MERA on a Large-Scale Example

In this experiment, we compare the compression capability between a TT and a MERA. We also apply Algorithm 2 on a large-scale example for which a 12-way cubical tensor $\mathcal{A}$ of dimension 10 is generated that is exactly represented by a 2-layer MERA, where each of the isometries reduces $K = 2$ indices into 1 index $S = 5$. The first layer of the MERA coarse-grains 12 indices into 6 indices and each of the isometries in this layer is a $10 \times 10 \times 5$ tensor. The second layer of the MERA coarse-grains the remaining 6 indices of the first layer into 3 indices and therefore consists of $5 \times 5 \times 5$ isometries. The top tensor of the MERA is a 3-way cubical tensor with dimension 5. All isometries and disentanglers are initialized as random matrices, drawn from an standard normal distribution, which are then made orthogonal or orthonormal through a QR decomposition. The top tensor is also initialized

as a random matrix. A comparison of the TT and MERA in terms of how well they compress the original 10^{12} is given in Table 3. The corresponding TT has TT-ranks $R_2 = 10, R_3 = 100, R_4 = 50, R_5 = 500, R_6 = 250, R_7 = 2\,500, R_8 = 250, R_9 = 500, R_{10} = 50, R_{11} = 100, R_{12} = 10$ and needs 15 620 200 elements. This constitutes a saving in storage space of $10^{12}/15\,620\,200 = 6.40 \times 10^4$. The MERA on the other hand consists of 54 750 elements and this results in a saving of storage space of $10^{12}/54\,750 = 1.82 \times 10^7$. The MERA is therefore about 285 times smaller as the TT.

Using Algorithm 2 to convert the TT back into a MERA with an identical structure as the "true" MERA ($K = 2$ and $S = 5$) takes 32.74 seconds and results in a relative approximation error of 1.00. This large approximation error is explained by the truncated HOSVD (line 9) in Algorithm 2 step not being able to truncate the ranks without losing accuracy. Using Algorithm 2 to convert the TT back into a MERA and using Algorithm 3 for the disentangler computation takes 63.81 seconds. Setting the stopping criterion for Algorithm 3 to $\sigma_{R'}/\sigma_{R'+1} > 10^{13}$ guarantees that a tolerance of 10^{-12} can be used for the truncated HOSVD, thus obtaining a $K = 2, S = 5$ MERA with a relative approximation error of 1.16×10^{-13}. The low-rank approximation used in Algorithm 3 contained 5, 25, 25, 25, 5 terms for the five disentanglers in the first layer, respectively, and 5 terms for the three disentanglers in the second layer. The 63.81 seconds runtime was dominated by Algorithm 3 reducing $R_7 = 2\,500$ down to a rank of 25, which took 53.88 seconds. The remaining 10 seconds were spent in the reduction of the ranks $R_6 = R_8 = 250$ whereas the computation of all remaining tensors in the MERA took fractions of seconds.

7 Conclusions

This article has introduced two new algorithms for the conversion of a TT into a Tucker decomposition and a MERA. The computation of a MERA-layer was shown to consist of one HOSVD-step for the computation of the disentanglers and one truncated HOSVD-step for the computation of the isometries. Using HOSVD to compute disentanglers was shown to be sub-optimal in terms of reducing the rank and an iterative orthogonal Procrustes algorithm was proposed that is able to find rank-lowering disentanglers. Numerical experiments have demonstrated the efficacy of the proposed algorithms. The TT to Tucker decomposition algorithm was demonstrated to be fast compared to the conventional HOSVD algorithm and resulted in an improvement of storage complexity that was one order of magnitude smaller. The MERA was shown to have even more potential in storage complexity in an experiment involving a tensor that consisted of 10^{12} elements where a compression improvement of a factor 285 compared to a TT was observed. The effectiveness and limitations of the orthogonal Procrustes algorithm were also explored in numerical experiments. The exact conditions under which this orthogonal Procrustes converges to a disentangler that retrieves an exact minimal-rank solution are still a topic for future research.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Batselier, K. (Kim): Data to reproduce experiments in research article "meracle: constructive layer-wise conversion of a tensor train into a MERA" (2020). https://doi.org/10.4121/UUID:CB37D1B8-A505-46EB-8C42-FE819429624B. https://data.4tu.nl/repository/uuid:cb37d1b8-a505-46eb-8c42-fe819429624b
2. Carroll, J., Chang, J.J.: Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. Psychometrika **35**(3), 283–319 (1970)
3. Cichocki, A., Lee, N., Oseledets, I., Phan, A.H., Zhao, Q., Mandic, D.P.: Tensor networks for dimensionality reduction and large-scale optimization: part 1 low-rank tensor decompositions. Foundations and Trends® in Machine Learning **9**(4/5), 249–429 (2016)
4. Cichocki, A., Mandic, D., De Lathauwer, L., Zhou, G., Zhao, Q., Caiafa, C., Phan, H.A.: Tensor decompositions for signal processing applications: from two-way to multiway component analysis. IEEE Sig. Process. Mag. **32**(2), 145–163 (2015)
5. Cichocki, A., Phan, A.H., Zhao, Q., Lee, N., Oseledets, I., Sugiyama, M., Mandic, D.P.: Tensor networks for dimensionality reduction and large-scale optimization: part 2 applications and future perspectives. Foundations and Trends® in Machine Learning **9**(6), 431–673 (2017)
6. De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. **21**(4), 1253–1278 (2000)
7. Dolgov, S., Khoromskij, B.: Two-level QTT-Tucker format for optimized tensor calculus. SIAM J. Matrix Anal. Appl. **34**(2), 593–623 (2013)
8. Espig, M., Hackbusch, W., Handschuh, S., Schneider, R.: Optimization problems in contracted tensor networks. Comput. Visualization Sci. **14**(6), 271–285 (2011)
9. Espig, M., Naraparaju, K.K., Schneider, J.: A note on tensor chain approximation. Comput. Visualization Sci. **15**(6), 331–344 (2012)
10. Evenbly, G., Vidal, G.: Algorithms for entanglement renormalization. Phys. Rev. B **79**, 144108 (2009)
11. Golub, G.H., van Loan, C.F.: Matrix Computations, fourth edn. Johns Hopkins University Press (2013)
12. Grasedyck, L.: Hierarchical singular value decomposition of tensors. SIAM J. Matrix Anal. Appl. **31**(4), 2029–2054 (2010)
13. Hackbusch, W., Kühn, S.: A new scheme for the tensor representation. J. Fourier Anal. Appl. **15**(5), 706–722 (2009)
14. Halko, N., Martinsson, P.G., Tropp, J.A.: Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev. **53**(2), 217–288 (2011)
15. Harshman, R.A.: Foundations of the PARAFAC procedure: models and conditions for an "explanatory" multi-modal factor analysis. UCLA Working Papers in Phonetics **16**(1), 84 (1970)
16. Hitchcock, F.: The expression of a tensor or a polyadic as a sum of products. J. Math. Phys. **6**, 164–189 (1927)
17. Holtz, S., Rohwedder, T., Schneider, R.: The alternating linear scheme for tensor optimization in the tensor train format. SIAM J. Sci. Comput. **34**(2), A683–A713 (2012)
18. Khoromskij, B.N.: $O(d\log N)$-quantics approximation of N-d tensors in high-dimensional numerical modeling. Constructive Approx. **34**(2), 257–280 (2011)
19. Kolda, T., Bader, B.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009)
20. Lehoucq, R.B., Sorensen, D.C.: Deflation techniques for an implicitly restarted Arnoldi iteration. SIAM J. Matrix Anal. Appl. **17**(4), 789–821 (1996)
21. Oseledets, I.: Tensor-train decomposition. SIAM J. Sci. Comput. **33**(5), 2295–2317 (2011)
22. Oseledets, I., Tyrtyshnikov, E.: TT-cross approximation for multidimensional arrays. Linear Algebra Appl. **422**(1), 70–88 (2010)
23. Rommer, S., Östlund, S.: Class of ansatz wave functions for one-dimensional spin systems and their relation to the density matrix renormalization group. Phys. Rev. B **55**, 2164–2181 (1997)
24. Schollwöck, U.: The density-matrix renormalization group in the age of matrix product states. Annals of Physics **326**(1), 96–192 (2011)

25. Shi, Y.Y., Duan, L.M., Vidal, G.: Classical simulation of quantum many-body systems with a tree tensor network. Phys. Rev. A **74**(2), 022320 (2006)
26. Sidiropoulos, N.D., De Lathauwer, L., Fu, X., Huang, K., Papalexakis, E.E., Faloutsos, C.: Tensor decomposition for signal processing and machine learning. IEEE Trans. Sig. Process. **65**(13), 3551–3582 (2017)
27. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika **31**(3), 279–311 (1966)
28. Vannieuwenhoven, N., Vandebril, R., Meerbergen, K.: A new truncation strategy for the higher-order singular value decomposition. SIAM J. Sci. Comput. **34**(2), A1027–A1052 (2012)
29. Vervliet, N., Debals, O., Sorber, L., Van Barel, M., De Lathauwer, L.: Tensorlab 3.0 (2016). https://www.tensorlab.net.
30. Vidal, G.: A class of quantum many-body states that can be efficiently simulated. Phys. Rev. Lett. **101**, 110501 (2008)