Alternating linear scheme in a bayesian framework for low-rank tensor approximation

Menzen, Clara; Kok, Manon; Batselier, Kim

DOI
10.1137/20M1386414

Publication date
2022

Document Version
Final published version

Published in
SIAM Journal on Scientific Computing

SIAM J. SCI. COMPUT.
Vol. 44, No. 3, pp. A1116–A1144

ALTERNATING LINEAR SCHEME IN A BAYESIAN FRAMEWORK FOR LOW-RANK TENSOR APPROXIMATION*

CLARA MENZEN†, MANON KOK†, AND KIM BATSELIER†

Abstract. Multiway data often naturally occurs in a tensorial format which can be approximately represented by a low-rank tensor decomposition. This is useful because complexity can be significantly reduced and the treatment of large-scale data sets can be facilitated. In this paper, we find a low-rank representation for a given tensor by solving a Bayesian inference problem. This is achieved by dividing the overall inference problem into subproblems where we sequentially infer the posterior distribution of one tensor decomposition component at a time. This leads to a probabilistic interpretation of the well-known iterative algorithm alternating linear scheme (ALS). In this way, the consideration of measurement noise is enabled, as well as the incorporation of application-specific prior knowledge and the uncertainty quantification of the low-rank tensor estimate. To compute the low-rank tensor estimate from the posterior distributions of the tensor decomposition components, we present an algorithm that performs the unscented transform in tensor train format.

Key words. low-rank approximation, alternating linear scheme, Bayesian inference, tensor decomposition, tensor train

AMS subject classifications. 15A69, 93E24, 15A23, 90C06, 62C10

DOI. 10.1137/20M1386414

1. Introduction.

Low-rank approximations of multidimensional arrays, also called tensors, have become a central tool in solving large-scale problems. The numerous applications include machine learning (e.g., tensor completion [36, 43, 14], kernel methods [35, 5] and deep learning [10, 25]), signal processing [34, 3], probabilistic modeling [21, 40], nonlinear system identification [13, 1], and solving linear systems [28, 12]. An extensive overview of applications can be found, e.g., in [9].

In many applications, an observed tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ can be represented with a low-rank approximation $\boldsymbol{\mathcal{Y}}_{\text{lr}}$, without losing the most meaningful information [8]. In the presence of uncorrelated noise $\boldsymbol{\mathcal{E}}$, however, $\boldsymbol{\mathcal{Y}}$ loses the low-rank structure. The observed tensor can be modeled as

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{Y}}_{\text{lr}}(\boldsymbol{\mathcal{G}}_1, \boldsymbol{\mathcal{G}}_2, \ldots, \boldsymbol{\mathcal{G}}_N) + \boldsymbol{\mathcal{E}}, \qquad \text{vec}(\boldsymbol{\mathcal{E}}) \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}), \tag{1.1}$$

where $\boldsymbol{\mathcal{Y}}_{\text{lr}}$ is a low-rank tensor decomposition (TD), which is a function of TD components $\boldsymbol{\mathcal{G}}_1, \boldsymbol{\mathcal{G}}_2, \ldots, \boldsymbol{\mathcal{G}}_N$. Examples of TDs are the CANDECOMP/PARAFAC (CP) decomposition [4, 15], the Tucker decomposition [37], and the tensor train (TT) decomposition [27]. We model the vectorized noise $\text{vec}(\boldsymbol{\mathcal{E}})$ as Gaussian, where $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ denotes a zero mean multivariate normal distribution with covariance matrix $\sigma^2\mathbf{I}$. The identity matrix is denoted by $\mathbf{I}$ which in (1.1) is of size $I_1 I_2 \ldots I_N \times I_1 I_2 \ldots I_N$.

In this work, we solve a Bayesian inference problem to seek a low-rank TD $\boldsymbol{\mathcal{Y}}_{\text{lr}}$, given an observed noisy tensor $\boldsymbol{\mathcal{Y}}$. In general, TDs solve an optimization problem of the form

$$\min_{\boldsymbol{\mathcal{G}}_1, \boldsymbol{\mathcal{G}}_2, \ldots, \boldsymbol{\mathcal{G}}_N} ||\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{Y}}_{\text{lr}}(\boldsymbol{\mathcal{G}}_1, \boldsymbol{\mathcal{G}}_2, \ldots, \boldsymbol{\mathcal{G}}_N)||. \tag{1.2}$$

*Submitted to the journal's Methods and Algorithms for Scientific Computing section December 18, 2020; accepted for publication (in revised form) November 11, 2021; published electronically May 5, 2022.

https://doi.org/10.1137/20M1386414

†Delft Center for Systems and Control, TU Delft, Delft, 2628 CD, The Netherlands (c.m.menzen@tudelft.nl, m.kok-1@tudelft.nl, k.batselier@tudelft.nl).

There exist multiple methods to find a decomposition for a given tensor. The approach that we are looking at in this paper is the well-known iterative method alternating linear scheme (ALS). The ALS has been studied extensively and successfully been applied to find low-rank tensor decompositions (TDs). The ALS for the CP decomposition is described in [23, 11], the Tucker decomposition is also treated in [23], and the ALS for the TT decomposition is studied in [20, 31]. The ALS optimizes the sought tensor on a manifold with fixed ranks [31, p. 1136]. Imposing the low-rank constraint is therefore easy to implement by choosing the ranks in advance.

The CP, Tucker, and TT decomposition are all multilinear functions of all the TD components. This means that by assuming all TD components except the nth to be known, the tensor becomes a linear expression in the nth component [11, p. 4]. In the ALS all TD components are updated sequentially by making use of the TD's multilinearity. Each update step requires solving a linear least squares problem given by

$$\min_{\mathbf{g}_n} ||\mathbf{y} - \mathbf{U}_{\setminus n}\mathbf{g}_n||_{\mathrm{F}}, \tag{1.3}$$

where $\mathbf{y} \in \mathbb{R}^{I_1 I_2 \dots I_N \times 1}$ and $\mathbf{g}_n \in \mathbb{R}^{K \times 1}$ denote the vectorization of $\boldsymbol{\mathcal{Y}}$ and $\boldsymbol{\mathcal{G}}_n$, respectively, K being the number of elements in the nth TD component. The matrix $\mathbf{U}_{\setminus n} \in \mathbb{R}^{J \times K}$ is a function of all TD components except the nth, where J is the number of elements in $\mathbf{y}$, and $||\cdot||_{\mathrm{F}}$ denotes the Frobenius norm.

A drawback of the ALS is that it does not explicitly model the measurement noise $\boldsymbol{\mathcal{E}}$, which in real-life applications is usually present. In this work, we model the noise by approaching the TD in a Bayesian framework, treating all components as probability distributions. In this way, finding a low-rank TD approximation can be solved as a Bayesian inference problem: given the prior distributions of the TD components $p(\mathbf{g}_i)$ and the measurements $\mathbf{y}$, the posterior distribution $p(\{\mathbf{g}_i\} \mid \mathbf{y})$ can be found by applying Bayes' rule:

$$p\left(\{\mathbf{g}_i\} \mid \mathbf{y}\right) = \frac{\overbrace{p(\mathbf{y}|\{\mathbf{g}_i\})}^{\text{likelihood}}\overbrace{p(\{\mathbf{g}_i\})}^{\text{prior}}}{\underbrace{p(\mathbf{y})}_{\text{evidence}}}, \tag{1.4}$$

where $\{\mathbf{g}_i\}$ denotes the collection of all TD components $\mathbf{g}_i$ for $i = 1, \dots, N$. We assume that the prior is Gaussian and, as in the ALS, we apply a block coordinate descent [24, p. 230]. This leads to a tractable inference for each substep.

Solving the low-rank tensor approximation problem in a Bayesian way has the following benefits. The assumptions on the measurement noise $\boldsymbol{\mathcal{E}}$ are considered and the uncertainty of each TD component $\mathbf{g}_n$ is quantified. Furthermore, prior knowledge can be explicitly taken into account and the resulting low-rank tensor estimate comes with a measure of uncertainty. We illustrate the benefits with numerical experiments.

Our main contribution is to approach the low-rank tensor approximation problem from a Bayesian perspective, treating all TD components as Gaussian random variables. This results in a probabilistic ALS algorithm. We ensure numerical stability by incorporating the orthogonalization step, present in the ALS algorithm for the TT decomposition, into the probabilistic framework. In addition, we propose an algorithm to approximate the mean and covariance of the low-rank tensor estimate's posterior density with the unscented transform in tensor train format. Our open-source MATLAB implementation can be found on https://gitlab.tudelft.nl/cmmenzen/bayesian-als.

Related work. Our work is related to inferring low-rank TDs with Bayesian methods for noisy continuous-valued multidimensional observations. While most literature considers either the CP or Tucker decomposition, our paper mainly focuses on the TT decomposition, but is also applicable to CP and Tucker. Also, in contrast to our paper, the related work mainly treats tensors with missing values. The main difference from the existing literature, however, is the modeling choices. While the existing work proposes different methods to perform approximate inference of the TD components, we make approximations that allow us to have a tractable inference. This is because we take inspiration from the ALS and each substep of the ALS has an analytical solution. Also, we assume that all TD components are Gaussian random variables and that they are all independent. Thus, our method is preferable when these assumptions can be made for a given application.

In [29, 30, 38] inference is performed with Gibbs sampling, using Gaussian priors for the columns of the CP decomposition's factor matrices. Variational Bayes is applied in [41, 43]. The recovery of orthogonal factor matrices, optimizing on the Stiefel manifold with variational inference, is treated by [6]. The Bayesian treatment of a low-rank Tucker decomposition for continuous data has been studied using variational inference [7, 42] and using Gibbs sampling [19]. Furthermore, an infinite Tucker decomposition based on a t-process, which is a kernel-based nonparametric Bayesian generalization of the low-rank Tucker decomposition, is proposed by [39]. The first literature about the probabilistic treatment of the TT decomposition using von-Mises–Fisher priors on the orthogonal cores and variational approximation with evidence lower bound is introduced by [18]. Recently, the authors of [17] published the probabilistic TD toolbox for MATLAB, providing inference with variational Bayes and with Gibbs sampling.

2. Tensor basics and notation. An N-way tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is a generalization of a vector or a matrix to higher dimensions, where N is often referred to as the order of the tensor. We denote tensors by calligraphic, boldface, capital letters (e.g., $\mathcal{Y}$) and matrices, vectors, and scalars by boldface capital (e.g., $\mathbf{Y}$), boldface lowercase (e.g., $\mathbf{y}$), and italic lower case (e.g., y) letters, respectively. To facilitate the description and computation of tensors, we use a graphical notation as depicted in Figure 1. The nodes represent a scalar, a vector, a matrix, and an N-way tensor and edges correspond to a specific index. The number of edges is equal to how many indices need to be specified to identify one element in the object, e.g., row and column index for matrices. An identity matrix is generally denoted by $\mathbf{I}$. Its size is either specified in the context or as a subscript.

Often it is easier to avoid working with the tensors directly, but rather with a matricized or vectorized version of them. Therefore, we revise some useful definitions. In this context, a mode of a tensor refers to a dimension of the tensor.

DEFINITION 2.1 (mode-n-unfolding [23, pp. 459–460]). *The transformation of an N-way tensor into a matrix with respect to a specific mode is called the mode-n unfolding. It is denoted by*

$$\mathbf{Y}_{(n)} \in \mathbb{R}^{I_n \times I_1 \ldots I_{n-1} I_{n+1} \ldots I_N}.$$

The vectorization is a special case of the unfolding, denoted by the operator name vec() and defined as

$$\text{vec}(\mathcal{Y}) = \mathbf{y} \in \mathbb{R}^{I_1 I_2 \ldots I_N \times 1}.$$

Tensors can be multiplied with matrices defined as follows.

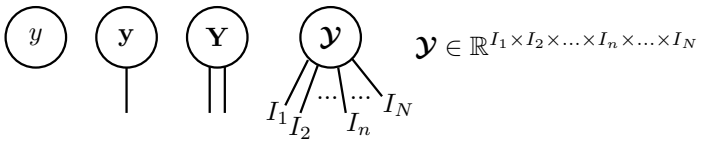

FIG. 1. *Visual depictions of a scalar, a vector, a matrix, and an N-way tensor, where the nodes represent the object and the edges correspond to a specific index. The number of edges is equal to how many indices need to be specified to identify one element in the object, e.g., row and column index for matrices.*

DEFINITION 2.2 (n-mode product [23, p. 460]). *The n-mode product is defined as the multiplication of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times \cdots \times I_N}$ with a matrix $\mathbf{A} \in \mathbb{R}^{J \times I_n}$ in mode n, written as*

$$\boldsymbol{\mathcal{X}} \times_n \mathbf{A} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}.$$

Elementwise, the $(i_1, i_2, \ldots, i_{n-1}, j, i_{n+1}, \ldots, i_N)$th entry of the result can be computed as

$$\sum_{i_n=1}^{I_n} \boldsymbol{\mathcal{X}}(i_1, i_2, \ldots, i_N)\mathbf{A}(j, i_n).$$

DEFINITION 2.3 (Kronecker product [23, p. 461]). *The Kronecker product of matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$ is denoted by $\mathbf{A} \otimes \mathbf{B}$. The result is a matrix of size $(KI) \times (LJ)$ and is defined by*

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \cdots & a_{IJ}\mathbf{B} \end{bmatrix}.$$

DEFINITION 2.4 (Khatri–Rao product [23, p. 462]). *The Khatri–Rao product of matrices $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$ is denoted by $\mathbf{A} \odot \mathbf{B}$. The result is a matrix of size $(JI) \times K$ defined by*

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_K \otimes \mathbf{b}_K \end{bmatrix}.$$

The visual depictions of two important matrix operations are shown in Figure 2. On the left, a product between matrices $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{K \times J}$ is shown, where the summation over the middle index K, also called contraction, is represented as an edge that connects both nodes. On the right, an outer product between matrices $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ and $\mathbf{B} \in \mathbb{R}^{J_1 \times J_2}$ is shown, where the dotted line represents a rank-1 connection. The resulting matrix is the Kronecker product $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{I_1 J_1 \times I_2 J_2}$.

A tensor can be expressed as a function of simpler tensors that form a TD. An extensive review about TDs can be found in [23]. The most notable are the CP decomposition, the Tucker decomposition, and the TT decomposition.

DEFINITION 2.5 (CP decomposition [4, 15]). *The CP decomposition consists of a set of matrices $\mathbf{G}_i \in \mathbb{R}^{I_i \times R}$, $i = 1, \ldots, N$, called factor matrices, and a weight vector $\boldsymbol{\lambda} \in \mathbb{R}^{R \times 1}$ that represent a given N-way tensor $\boldsymbol{\mathcal{Y}}$. Elementwise, the $(i_1, i_2, \ldots, i_N)$th entry of $\boldsymbol{\mathcal{Y}}$ can be computed as*

$$\sum_{r=1}^{R} \boldsymbol{\lambda}(r)\mathbf{G}_1(i_1, r) \cdots \mathbf{G}_N(i_N, r),$$

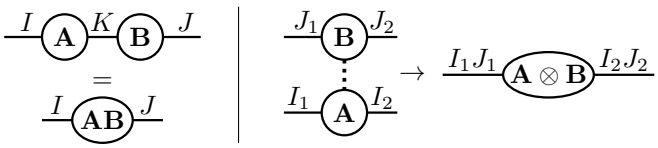

FIG. 2. *Left: Visual depictions of an index contraction between matrices* $\mathbf{A}$ *and* $\mathbf{B}$. *Right: Visual depictions of an outer product between matrices* $\mathbf{A}$ *and* $\mathbf{B}$. *The dotted line represents a summation over a rank-*1 *one connection. The resulting matrix is computed as the Kronecker product* $\mathbf{A} \otimes \mathbf{B}$.

where R *denotes the rank of the decomposition.*

DEFINITION 2.6 (Tucker decomposition [37]). *The Tucker decomposition consists of an* N*-way tensor* $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$*, called core tensor, and a set of matrices* $\mathbf{G}_i \in \mathbb{R}^{I_i \times R_i}$*,* $i = 1, \ldots, N$*, called factor matrices, that represent a given* N*-way tensor* $\boldsymbol{\mathcal{Y}}$*. Elementwise, the* $(i_1, i_2, \ldots, i_N)$*th entry of* $\boldsymbol{\mathcal{Y}}$ *can be computed as*

$$\sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} \boldsymbol{\mathcal{C}}(r_1, \ldots, r_N) \mathbf{G}_1(i_1, r_1) \cdots \mathbf{G}_N(i_N, r_N),$$

where $R_1, \ldots, R_N$ *denote the ranks of the decomposition. The factor matrices can be orthogonal, such that the Frobenius norm of the entire tensor is contained in the core tensor.*

DEFINITION 2.7 (the TT decomposition [27]). *The TT decomposition consists of a set of three-way tensors* $\boldsymbol{\mathcal{G}}_i \in \mathbb{R}^{R_i \times I_i \times R_{i+1}}$*,* $i = 1, \ldots, N$*, called TT-cores, that represent a given* N*-way tensor* $\boldsymbol{\mathcal{Y}}$*. Elementwise, the* $(i_1, i_2, \ldots, i_N)$*th entry of* $\boldsymbol{\mathcal{Y}}$ *can be computed as*

$$\sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_{N+1}=1}^{R_{N+1}} \boldsymbol{\mathcal{G}}_1(r_1, i_1, r_2) \boldsymbol{\mathcal{G}}_2(r_2, i_2, r_3) \cdots \boldsymbol{\mathcal{G}}_N(r_N, i_N, r_{N+1}),$$

where $R_1, \ldots, R_{N+1}$ *denote the ranks of the TT-cores and by definition* $R_1 = R_{N+1} = 1$.

If the tensor is only approximately represented by a TD, then the ranks determine the accuracy of the approximation.

As mentioned in section 1, to formulate the linear least squares problem for one update of the ALS, the TD's property of multilinearity is exploited and it is expressed as $\mathbf{y} = \mathbf{U}_{\setminus n} \mathbf{g}_n$, with $\mathbf{U}_{\setminus n} \in \mathbb{R}^{J \times K}$ and $\mathbf{g}_n \in \mathbb{R}^{K \times 1}$, where J and K are the number of elements of $\boldsymbol{\mathcal{Y}}$ and $\boldsymbol{\mathcal{G}}_n$, respectively. The following three examples describe how $\mathbf{U}_{\setminus n}$ is built for the CP decomposition, the Tucker decomposition, and the TT decomposition.

Example 2.8. If a tensor is represented in terms of a CP decomposition, the matrix $\mathbf{U}_{\setminus n}$ can be written as

$$\mathbf{U}_{\setminus n} = (\mathbf{G}_N \odot \cdots \odot \mathbf{G}_{n+1} \odot \mathbf{G}_{n-1} \odot \cdots \odot \mathbf{G}_1) \otimes \mathbf{I}_{I_n}. \tag{2.1}$$

Note that $\mathbf{U}_{\setminus n}$ is of size $I_n I_1 \ldots I_{n-1} I_{n+1} \ldots I_N \times I_n R$, so the first dimension needs to be permuted in order to match $\mathbf{y} \in \mathbb{R}^{I_1 I_2 \ldots I_N}$. The weight vector $\boldsymbol{\lambda}$ is absorbed into the factor matrix that is being updated. After each update, the columns of $\mathbf{G}_n$ are normalized and the norms are stored in $\boldsymbol{\lambda}$. The CP-ALS algorithm can be found in [23, p. 471].

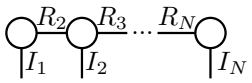

FIG. 3. *Visual depiction of a TT decomposition with N TT-cores.*

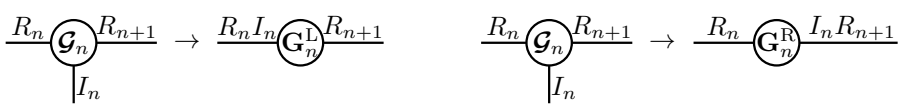

FIG. 4. *Left: Visual depiction of a left-unfolding of a TT-core. Right: Right-unfolding of a TT-core.*

Example 2.9. If a tensor is represented in terms of a Tucker decomposition, the matrix $\mathbf{U}_{\backslash n}$ can be written as

$$\mathbf{U}_{\backslash n} = \left[\left(\mathbf{G}_N \otimes \cdots \otimes \mathbf{G}_{n+1} \otimes \mathbf{G}_{n-1} \otimes \ldots \mathbf{G}_1 \right) \mathbf{C}_{(n)}^{\top} \right] \otimes \mathbf{I}_{I_n}. \tag{2.2}$$

Note that $\mathbf{U}_{\backslash n}$ is of size $I_n I_1 \ldots I_{n-1} I_{n+1} \ldots I_N \times I_n R_n$, so the first dimension needs to be permuted in order to match $\mathbf{y} \in \mathbb{R}^{I_1 I_2 \ldots I_N}$. The core tensor is recomputed by solving

$$\mathbf{y} = \left(\mathbf{G}_N \otimes \cdots \otimes \mathbf{G}_1 \right) \mathrm{vec}(\boldsymbol{\mathcal{C}}).$$

Example 2.10. If the tensor is represented in terms of a TT decomposition, the matrix $\mathbf{U}_{\backslash n}$ can be written as

$$\mathbf{U}_{\backslash n} = \boldsymbol{\mathcal{G}}_{i>n} \otimes \mathbf{I}_{I_n} \otimes \boldsymbol{\mathcal{G}}_{i<n}^{\top} \in \mathbb{R}^{I_1 I_2 \ldots I_N \times R_n I_n R_{n+1}}, \tag{2.3}$$

where $\boldsymbol{\mathcal{G}}_{i<n}$ ($\boldsymbol{\mathcal{G}}_{i>n}$) denotes a tensor obtained by contracting the TD components, left (right) of the nth core.

From here on, we will focus on the TT decomposition. We, therefore, review some of the main concepts. A TT can be represented by a diagram with nodes as the TT-cores and the edges as the modes of the approximated tensor. Connected edges are the summation over the ranks between two cores (Figure 3). To introduce a notion of orthonormality for TT-cores, a special case of Definition 2.1 is used, creating unfoldings of the TT-cores defined as follows.

DEFINITION 2.11 (left- and right-unfolding [20, p. A689]). *The left-unfolding* $\mathbf{G}_n^{\mathrm{L}}$ *and right-unfolding* $\mathbf{G}_n^{\mathrm{R}}$ *of a TT-core* $\boldsymbol{\mathcal{G}}_n$ *are the unfoldings of a core with respect to the first and last mode, respectively (Figure* 4*). Please note that the definition by* [20] *of the right-unfolding is the transposed version of this definition.*

DEFINITION 2.12 (left-orthogonal and right-orthogonal [20, p. A689]). *A TT-core* $\boldsymbol{\mathcal{G}}_n$ *is called left-orthogonal if the left-unfolding* $\mathbf{G}_n^{\mathrm{L}}$ *satisfies*

$$\left(\mathbf{G}_n^{\mathrm{L}} \right)^{\top} \mathbf{G}_n^{\mathrm{L}} = \mathbf{I}_{R_{n+1}}.$$

Analogously, a TT-core $\boldsymbol{\mathcal{G}}_n$ *is called right-orthogonal if the right-unfolding* $\mathbf{G}_n^{\mathrm{R}}$ *satisfies*

$$\mathbf{G}_n^{\mathrm{R}} \left(\mathbf{G}_n^{\mathrm{R}} \right)^{\top} = \mathbf{I}_{R_n}.$$

DEFINITION 2.13 (site-n-mixed-canonical form [33, p. 113]). *A TT is in site-*n*-mixed-canonical form if the TT-cores* $\{\boldsymbol{\mathcal{G}}_i\}_{i<n}$ *are left-orthogonal and the TT-cores*

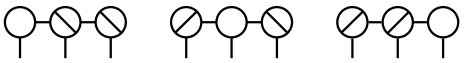

FIG. 5. *Visual depiction of TTs with three TT-cores in site-n-mixed-canonical form. Left: Norm in the first core, and other cores left-orthogonal. Middle: Norm in the second core, and first and last cores are left- and right-orthogonal, respectively. Right: Norm in the last core, and other cores right-orthogonal.*

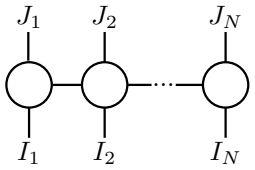

FIG. 6. *Visual depiction of a TT matrix. The row indices $I_1, \ldots, I_N$ point downwards, and the column indices $J_1, \ldots, J_N$ point upwards.*

$\{\boldsymbol{\mathcal{G}}_i\}_{i>n}$ are right-orthogonal. The nth TT-core is not orthogonal, and it can be easily shown that

$$||\boldsymbol{\mathcal{Y}}||_{\mathrm{F}} = ||\boldsymbol{\mathcal{G}}_n||_{\mathrm{F}}.$$

Figure 5 depicts different site-n-mixed-canonical forms for an exemplary three-way TT. In the left (right) figure, the Frobenius norm is contained in the first (last) core and all other cores are right- (left-) orthogonal, represented by the diagonal in the node.

A special case of the TT decomposition format is the TT matrix (Figure 6), which represents a large matrix in TT format. TT matrices arise in the context of the unscented transform in section 5.

DEFINITION 2.14 (TT matrix [26]). *A TT matrix (TTm) consists of a set of four-way tensors $\boldsymbol{\mathcal{G}}_i \in \mathbb{R}^{R_i \times I_i \times J_i \times R_{i+1}}$, $i = 1, \ldots, N$ with $R_1 = R_{N+1} = 1$, that represents a matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$. The row and column indices are split into multiple row indices $I = I_1, \ldots, I_N$ and column indices $J = J_1, \ldots, J_N$, respectively, and the matrix is transformed into a $2N$-way tensor $\boldsymbol{\mathcal{Y}}_{\mathbf{A}} \in \mathbb{R}^{I_1 \times J_1 \times \cdots \times I_N \times J_N}$. Elementwise, the $(i_1, j_1, i_2, j_2, \ldots, i_N, j_N)$th entry of $\boldsymbol{\mathcal{Y}}_{\mathbf{A}}$ is computed as*

$$\sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_{N+1}=1}^{R_{N+1}} \boldsymbol{\mathcal{G}}_1(r_1, i_1, j_1, r_2) \boldsymbol{\mathcal{G}}_2(r_2, i_2, j_2, r_3) \cdots \boldsymbol{\mathcal{G}}_N(r_N, i_N, j_N, r_{N+1}).$$

A TTm arises, e.g., from an outer product between two vectors $\mathbf{a}$ and $\mathbf{b}$, which corresponds to computing the product of one vector with the transpose of the other. If vector $\mathbf{a}$ is represented by a TT with cores $\boldsymbol{\mathcal{A}}_1, \ldots, \boldsymbol{\mathcal{A}}_N$, the resulting TTm is achieved by summing over a rank-1 connection between one of the TT-cores, e.g., the first, and vector $\mathbf{b}$ (Figure 7 top). This result is a special case of the general TTm, where only one of the TT-cores has a double index. This means that only the row index is very large and therefore split into multiple indices, while the column index is not split. If both vectors in the outer product are represented by TTs with cores $\boldsymbol{\mathcal{A}}_1, \ldots, \boldsymbol{\mathcal{A}}_N$ and $\boldsymbol{\mathcal{B}}_1, \ldots, \boldsymbol{\mathcal{B}}_N$, respectively, then each core is summed over a rank-1 connection with the core of the other TT's transpose (Figure 7, middle). All cores then have a row and column indices. The product of a matrix $\mathbf{C}$ in TTm format with cores $\boldsymbol{\mathcal{C}}_1, \ldots, \boldsymbol{\mathcal{C}}_N$

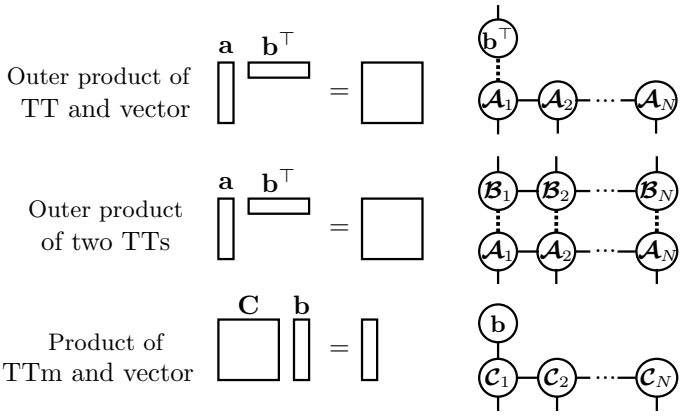

FIG. 7. *Operations with matrices and vectors. Top: Outer product between a vector* $\mathbf{a}$*, represented by a TT with cores* $\boldsymbol{\mathcal{A}}_1, \ldots, \boldsymbol{\mathcal{A}}_N$*, and a vector* $\mathbf{b}$*. A rank-*1 *connection (dotted line) is summed over between the first TT-core and vector* $\mathbf{b}^\top$*. Middle: Outer product between two vectors* $\mathbf{a}$ *and* $\mathbf{b}$ *represented by TTs with cores* $\boldsymbol{\mathcal{A}}_1, \ldots, \boldsymbol{\mathcal{A}}_N$ *and* $\boldsymbol{\mathcal{B}}_1, \ldots, \boldsymbol{\mathcal{B}}_N$*, respectively. A rank-*1 *connection is summed over between each core of the TTs. Bottom: The product between a matrix* $\mathbf{C}$ *in TTm format with cores* $\boldsymbol{\mathcal{C}}_1, \ldots, \boldsymbol{\mathcal{C}}_N$ *and a vector* $\mathbf{b}$*. The column index of the first TTm-core is summed over with the row index of the vector* $\mathbf{b}$*.*

with a vector $\mathbf{b}$ is computed by summing over the column index of one TTm-core, e.g., the first, and the row index of the vector (Figure 7, bottom).

3. Bayesian inference for low-rank tensor approximation. In this section, we present a method to find a low-rank TD using a similar strategy as in the ALS by solving a Bayesian inference problem. In this context, the vectorization of each TD component is treated as a Gaussian random variable, expressed in terms of a mean and a covariance. Generally, we denote a Gaussian probability distribution as $\mathcal{N}(\mathbf{m}, \mathbf{P})$, where $\mathbf{m}$ is the mean and $\mathbf{P}$ is the covariance. This section is organized as follows. First, we define the prior for the inference problem. Before computing the joint posterior distribution, we look at a simpler inference problem, stated in Lemma 3.1, where the posterior distribution of only one TD component is computed. Then, Theorem 3.3 describes the computation of the joint posterior by applying a block coordinate descent method and simplifying the inference problem to iteratively applying Lemma 3.1. Finally, our resulting Algorithm 3.1 is applied in an example.

To initialize the Bayesian inference problem, a multivariate Gaussian prior is assigned to every TD component

$$p(\mathbf{g}_i) = \mathcal{N}\left(\mathbf{m}_i^0, \mathbf{P}_i^0\right), \qquad i = 1, \ldots, N,$$

where $\mathbf{m}_i^0$ and $\mathbf{P}_i^0$ are the prior mean and covariance matrix, respectively. The TD components $\mathbf{g}_i \in \mathbb{R}^{R_i I_i R_{i+1} \times 1}$ are assumed to be statistically independent. Therefore, the joint prior distribution is given by

$$p(\{\mathbf{g}_i\}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{m}_1^0 \\ \mathbf{m}_2^0 \\ \vdots \\ \mathbf{m}_N^0 \end{bmatrix}, \begin{bmatrix} \mathbf{P}_1^0 & 0 & \ldots & 0 \\ 0 & \mathbf{P}_2^0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & \mathbf{P}_N^0 \end{bmatrix}\right),$$

where $\{\mathbf{g}_i\}$ denotes the priors of all TD components. Because of the statistical inde-

pendence, the joint prior distribution and the prior on one TD component conditioned on the other TD components can be written as

$$p(\{\mathbf{g}_i\}) = p(\mathbf{g}_1)p(\mathbf{g}_2)\dots p(\mathbf{g}_N) \quad \text{and} \tag{3.1}$$

$$p(\mathbf{g}_n \mid \{\mathbf{g}_i\}_{i\neq n}) = p(\mathbf{g}_n), \tag{3.2}$$

respectively, where $\{\mathbf{g}_i\}_{i\neq n}$ denotes the collection of all TD components except the nth.

The joint posterior distribution $p(\{\mathbf{g}_i\} \mid \mathbf{y})$ is found by applying Bayes' rule. However, before solving this inference problem and inspired by a result described in [32, p. 29], we first look at the simpler problem to find the posterior distribution of one component, given the measurement and the other components.

LEMMA 3.1. *Let the prior distribution $p(\mathbf{g}_n) = \mathcal{N}(\mathbf{m}_n^0, \mathbf{P}_n^0)$ and the likelihood $p(\mathbf{y} \mid \{\mathbf{g}_i\}) = \mathcal{N}(\mathbf{m}_\mathbf{y}, \sigma^2\mathbf{I})$ be Gaussian, where $\mathbf{m}_\mathbf{y} = \mathbf{U}_{\setminus n}\mathbf{g}_n$. Further, let all TD components be statistically independent, and let the TD be multilinear. Then, the posterior distribution $p(\mathbf{g}_n \mid \{\mathbf{g}_i\}_{i\neq n}, \mathbf{y}) = \mathcal{N}(\mathbf{m}_n^+, \mathbf{P}_n^+)$ of the nth component given the measurements and the other components is also Gaussian with mean $\mathbf{m}_n^+$ and covariance $\mathbf{P}_n^+$:*

$$\mathbf{m}_n^+ = \left[(\mathbf{P}_n^0)^{-1} + \frac{\mathbf{U}_{\setminus n}^\top \mathbf{U}_{\setminus n}}{\sigma^2}\right]^{-1} \left[\frac{\mathbf{U}_{\setminus n}^\top \mathbf{y}}{\sigma^2} + (\mathbf{P}_n^0)^{-1}\mathbf{m}_n^0\right], \tag{3.3}$$

$$\mathbf{P}_n^+ = \left[(\mathbf{P}_n^0)^{-1} + \frac{\mathbf{U}_{\setminus n}^\top \mathbf{U}_{\setminus n}}{\sigma^2}\right]^{-1}. \tag{3.4}$$

Proof. The posterior distribution of one TD component conditioned on the other TD components and the measurements $p(\mathbf{g}_n \mid \mathbf{y}, \{\mathbf{g}_i\}_{i\neq n})$ can be found by applying Bayes' rule. Assuming that all components are statistically independent (3.2) leads to

$$p(\mathbf{g}_n \mid \mathbf{y}, \{\mathbf{g}_i\}_{i\neq n}) = \frac{p(\mathbf{y} \mid \{\mathbf{g}_i\})p(\mathbf{g}_n)}{p(\mathbf{y} \mid \{\mathbf{g}_i\}_{i\neq n})}. \tag{3.5}$$

Since the likelihood $p(\mathbf{y} \mid \{\mathbf{g}_i\})$ and prior $p(\mathbf{g}_n)$ are Gaussian, also the posterior will be Gaussian [32, pp. 28–29, 209–210] with mean (3.3) and covariance (3.4). ∎

COROLLARY 3.2. *For $\lim \mathbf{P}_n^0 \to \infty$, (3.3) reduces to the normal equations of the least squares problem and therefore the update equation of the conventional ALS*

$$\mathbf{m}_n^+ = \left(\mathbf{U}_{\setminus n}^\top \mathbf{U}_{\setminus n}\right)^{-1} \mathbf{U}_{\setminus n}^\top \mathbf{y}. \tag{3.6}$$

Corollary 3.2 describes the case where there is no useful prior information available for the nth TD component. Thus, the certainty on the prior mean is zero, and $\lim \mathbf{P}_n^0 \to \infty$.

Now, we can use Lemma 3.1 to find the joint posterior distribution of all TD components as described in the following theorem.

THEOREM 3.3. *Let $p(\{\mathbf{g}_i\} \mid \mathbf{y})$ be the posterior joint distribution of all TD components given $\mathbf{y}$. Further, let the prior distribution $p(\mathbf{g}_n) = \mathcal{N}(\mathbf{m}_n^0, \mathbf{P}_n^0)$ of any component as well as the likelihood $p(\mathbf{y} \mid \{\mathbf{g}_i\}) = \mathcal{N}(\mathbf{m}_\mathbf{y}, \sigma^2\mathbf{I})$ be Gaussian, where the mean $\mathbf{m}_\mathbf{y}$ is the tensor represented by the TD, which is a nonlinear function of all the*

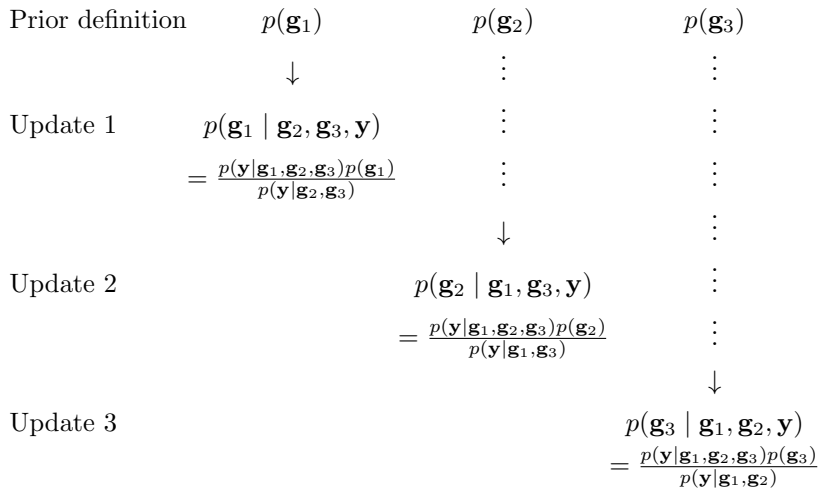

FIG. 8. *Distribution updates for example with three TD components.*

TD components. Further, let all TD components be statistically independent, and let the TD be multilinear. Then, by applying block coordinate descent to find the posterior density, every step of the block coordinate descent corresponds to applying Lemma 3.1.

Proof. Bayes' rule and statistical independence (3.1) gives

$$p\left(\{\mathbf{g}_i\} \mid \mathbf{y}\right) = \frac{p(\mathbf{y} \mid \{\mathbf{g}_i\})p(\mathbf{g}_1)p(\mathbf{g}_2)\dots p(\mathbf{g}_N)}{p(\mathbf{y})}. \tag{3.7}$$

As in the conventional ALS, a block coordinate descent method is applied by conditioning the posterior distribution of one TD component on all the others. In this way, the TD components can be computed sequentially with (3.5). In addition, due to the multilinearity of the TD the mean of the likelihood becomes a linear function of the nth TD component, $\mathbf{m}_\mathbf{y} = \mathbf{U}_{\setminus n}\mathbf{g}_n$. Thus, every TD update corresponds to applying Lemma 3.1. ∎

With Corollary 3.2, Theorem 3.3 gives a Bayesian interpretation of the ALS by deriving its update equation from the TD components defined as probability distributions. The following example shows how the distributions change with every update.

Example 3.4 (distribution updates for a TD with three components). Assume we would like to apply Theorem 3.3 to find a TD with three components. First, the three TD components are initialized with a prior distribution. Then, the distributions are updated sequentially by computing the posterior with Bayes' rule, as shown in Figure 8. After updating the three TD components, the updates are repeated until a stopping criterion is met.

Algorithm 3.1 summarizes the steps of the ALS in a Bayesian framework. The mean and covariance of each TD component are sequentially updated, followed by the computation of $\mathbf{U}_{\setminus n}$ which is computed from $\{\mathbf{g}_i\}_{i\neq n}$. The stopping criterion is defined by the user, e.g., as a maximum number of iterations or the convergence of the residuals between the measurement and estimate, as used in the convectional ALS. It is also possible to consider the convergence of the TT-core's covariance matrices as a stopping criterion since these are additionally computed in the ALS in the Bayesian

TABLE 1
Computational cost per update and overall storage requirements for Algorithm 3.1.

TD	Computational cost	Storage
CP	$\mathcal{O}(R^3I^3) + \mathcal{O}(RI^{N+1})$	$\mathcal{O}(NRI + NR^2I^2)$
Tucker	$\mathcal{O}(R^{3N}) + \mathcal{O}(R^N I^N)$	$\mathcal{O}(NRI + R^N + NR^2I^2 + R^{2N})$
TT	$\mathcal{O}(R^6I^3) + \mathcal{O}(R^2I^N)$	$\mathcal{O}(NR^2I + NR^4I^2)$

framework. Another possibility is to look at the convergence of the numerator of Bayes' rule.

For the complexity analysis, we use the following notation. The largest rank or the CP-rank is denoted by R, and the largest dimension of the approximated tensor is denoted by I. The computational cost per update and the overall storage requirements are given in Table 1. The first term in the computational cost for CP, Tucker, and TT represents the inversion that needs to be performed to compute the covariance matrix of the updated factor matrices, core tensor, and TT cores, respectively. The cost for the Tucker core could be reduced, however, by incorporating an orthogonalization step, thus, avoiding the recomputation of the core tensor. The second term for all TDs is the complexity to compute $\mathbf{U}_{\setminus n}^\top \mathbf{y}$, which is dominant compared to the cost of computing $\mathbf{U}_{\setminus n}^\top \mathbf{U}_{\setminus n}$. In comparison, the conventional ALS has the same computational cost for every TD component update, and a total storage requirement of $\mathcal{O}(NRI)$ for CP, $\mathcal{O}(NRI + R^N)$ for Tucker, and $\mathcal{O}(NR^2I)$ for TTs. The ALS in a Bayesian framework has an additional term in the storage requirements, because it computes the covariance matrix for every TD component.

Our method also opens up the possibility of recursively estimating the mean and covariance of the TD components. In case a new noisy measurement $\mathbf{y}$ of the same underlying tensor becomes available, Algorithm 3.1 can be applied repeatedly with the output mean and covariance from the previous execution as the prior for the new execution.

Algorithm 3.1. ALS in a Bayesian framework.

Require: Prior mean $\{\mathbf{m}_i^0\}$ and covariance $\{\mathbf{P}_i^0\}$, $i = 1, \dots N$, measurement $\mathbf{y}$, and noise variance σ^2.

Ensure: Posterior mean $\{\mathbf{m}_i^+\}$ and covariance $\{\mathbf{P}_i^+\}$, $i = 1, \dots N$.

Set $\{\mathbf{m}_i\} := \{\mathbf{m}_i^0\}$, $\{\mathbf{P}_i\} := \{\mathbf{P}_i^0\}$, $i = 1, \dots N$.
while Stopping criterion is not true **do**
  for $n = 1, \dots, N$ **do**
    Compute $\mathbf{U}_{\setminus n}$ with (2.1) for CP, (2.2) for Tucker or (2.3) for TT, using the mean of the TD components $\{\mathbf{m}_i\}_{i \neq n}$.
    $\mathbf{P}_n^+ \leftarrow \left[(\mathbf{P}_n^0)^{-1} + \frac{\mathbf{U}_{\setminus n}^\top \mathbf{U}_{\setminus n}}{\sigma^2} \right]^{-1}$
    $\mathbf{m}_n^+ \leftarrow \mathbf{P}_n^+ \left[\frac{\mathbf{U}_{\setminus n}^\top \mathbf{y}}{\sigma^2} + (\mathbf{P}_n^0)^{-1} \mathbf{m}_n^0 \right]$
    $\mathbf{m}_n \leftarrow \mathbf{m}_n^+$, $\mathbf{P}_n \leftarrow \mathbf{P}_n^+$
  end for
end while

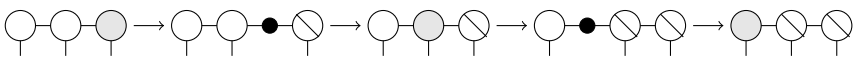

FIG. 9. *Visual depiction of a TT transformation into site-1-mixed-canonical form.*

4. Orthogonalization step in Bayesian framework for a TT. Every iteration of Algorithm 3.1 requires the inversion

$$\left[(\mathbf{P}_n^0)^{-1} + \frac{\mathbf{U}_{\setminus n}^\top \mathbf{U}_{\setminus n}}{\sigma^2}\right]^{-1}, \quad \text{corresponding to} \quad \left[\mathbf{U}_{\setminus n}^\top \mathbf{U}_{\setminus n}\right]^{-1} \tag{4.1}$$

in the conventional ALS update. To avoid the propagation of numerical errors and ensure numerical stability, some ALS algorithms, e.g., the one for the TT decomposition, include an orthogonalization step after every update. In this way, the condition number of each subproblem cannot become worse than the one of the overall problem [20, p. A701]. In this section, we present how we integrate the orthogonalization procedure into the ALS in a Bayesian framework for a TT decomposition in site-n-mixed-canonical form. The same can also be applied to a Tucker decomposition with orthogonal factor matrices.

We first describe how the orthogonalization step is performed in the conventional ALS and then how we integrate it into the ALS in a Bayesian framework. Here, we differentiate between the prior distributions of each TT-core and the initial guess for each TT-core, which initializes the conventional ALS. In the conventional ALS with orthogonalization step, the initial TT is transformed into the site-1-mixed-canonical form, where the Frobenius norm of the first TT-core corresponds to the Frobenius norm of the entire TT. The update is always performed on the core that contains the Frobenius norm. The procedure, therefore, requires transformations that separate the Frobenius norm from the updated TT-core and moves it to the next TT-core to be updated.

The initial TT is transformed into site-1-mixed-canonical form, by orthogonalizing the Nth up to the 2nd TT-core as illustrated in Figure 9 for a TT with three cores. To move the Frobenius norm from the nth TT-core to the $(n-1)$th, the nth TT-core is orthogonalized by applying the thin **QR**-decomposition on

$$\left(\mathbf{G}_n^{\mathrm{R}}\right)^\top = \mathbf{Q}_n^{\mathrm{R}} \mathbf{R}_n^{\mathrm{R}}. \tag{4.2}$$

Then, $\mathbf{G}_n^{\mathrm{R}}$ is replaced by

$$\mathbf{G}_n^{\mathrm{R}} \leftarrow \left(\mathbf{Q}_n^{\mathrm{R}}\right)^\top, \tag{4.3}$$

and the nonorthogonal part, illustrated by the small circle in Figure 9, is absorbed into the $(n-1)$th core with

$$\boldsymbol{\mathcal{G}}_{n-1} \leftarrow \boldsymbol{\mathcal{G}}_{n-1} \times_3 \mathbf{R}_n^{\mathrm{R}}. \tag{4.4}$$

Equations (4.2)–(4.4) are applied to the Nth until the 2nd TT-core, leading to the TT in site-1-mixed-canonical form. Then, the first core is updated, followed by a transformation to move the Frobenius norm to the second core, and so on. Since the Frobenius norm moves to the right, the orthogonalization step consists of applying the thin **QR**-decomposition on the left-unfolding

$$\mathbf{G}_n^{\mathrm{L}} = \mathbf{Q}_n^{\mathrm{L}} \mathbf{R}_n^{\mathrm{L}}. \tag{4.5}$$

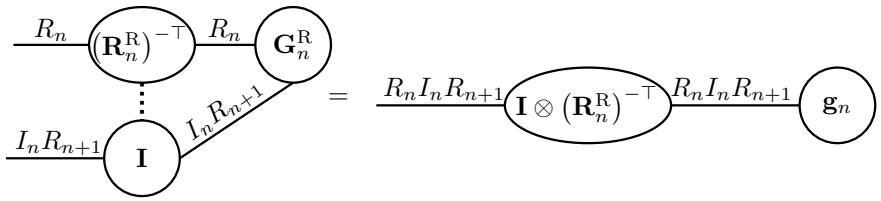

FIG. 10. *Visual depiction of how the nonorthogonal part is separated from the TD component's mean.*

The nth and $(n+1)$th core are replaced by

$$\mathbf{G}_n^{\mathrm{L}} \leftarrow \mathbf{Q}_n^{\mathrm{L}} \quad \text{and} \quad \boldsymbol{\mathcal{G}}_{n+1} \leftarrow \boldsymbol{\mathcal{G}}_{n+1} \times_1 \mathbf{R}_n^{\mathrm{L}}, \tag{4.6}$$

respectively. After the Nth core is updated, the updating scheme goes backwards, using again (4.2)–(4.4) for the orthogonalization step. When the Frobenius norm is absorbed back into the first core, one back-and-forth sweep of the ALS algorithm is concluded.

In the following, we describe how the transformation steps affect the distributions representing the TT-cores in the ALS in a Bayesian framework. The transformation of the random variables can be derived from (4.2)–(4.6). When the Frobenius norm is moved to the left, the mean of the nth core becomes

$$\mathbf{m}_n \leftarrow \operatorname{vec}\left(\left(\mathbf{Q}_n^{\mathrm{R}}\right)^{\top}\right),$$

where $\left(\mathbf{Q}_n^{\mathrm{R}}\right)^{\top}$ is computed from (4.3). To obtain the transformed covariance of the nth TT-core, (4.2) is rewritten as

$$\left(\mathbf{Q}_n^{\mathrm{R}}\right)^{\top} = \left(\mathbf{R}_n^{\mathrm{R}}\right)^{-\top} \mathbf{G}_n^{\mathrm{R}}.$$

Now, the right-hand side, is vectorized by summing over a rank-1 connection between $\left(\mathbf{R}_n^{\mathrm{R}}\right)^{-\top}$ and an identity matrix of size $I_n R_{n+1} \times I_n R_{n+1}$ that has a connected edge with $\mathbf{G}_n^{\mathrm{R}}$ as depicted in Figure 10. This leads to a transformation term

$$\mathbf{I} \otimes \left(\mathbf{R}_n^{\mathrm{R}}\right)^{-\top} \tag{4.7}$$

that orthogonalizes $\mathbf{g}_n$. The diagram in Figure 11 shows how this transformation is applied to the covariance matrix. The transformation term and its transpose are multiplied on the left and right side of $\mathbf{P}_n$, respectively, resulting in

$$\begin{aligned} \mathbf{P}_n \leftarrow & \left(\mathbf{I} \otimes \left(\mathbf{R}_n^{\mathrm{R}}\right)^{-\top}\right) \mathbf{P}_n \left(\mathbf{I} \otimes \left(\mathbf{R}_n^{\mathrm{R}}\right)^{-\top}\right)^{\top} \\ = & \left(\mathbf{I} \otimes \left(\mathbf{R}_n^{\mathrm{R}}\right)^{-\top}\right) \mathbf{P}_n \left(\mathbf{I} \otimes \left(\mathbf{R}_n^{\mathrm{R}}\right)^{-1}\right). \end{aligned}$$

The transformations of the $(n-1)$th core to absorb the Frobenius norm can be derived from (4.4) in a similar way as explained above, resulting in a transformation term

$$\mathbf{R}_n^{\mathrm{R}} \otimes \mathbf{I}. \tag{4.8}$$

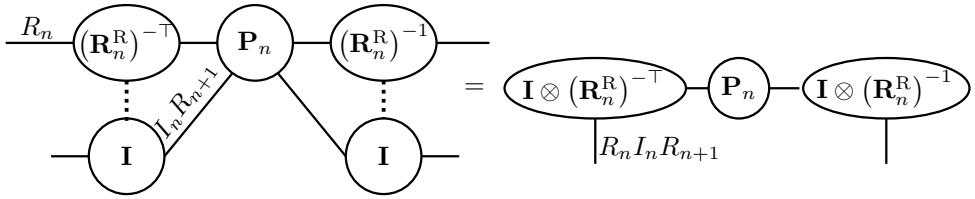

FIG. 11. *Visual depiction of how the covariance matrix is transformed in the orthogonalization step.*

When the Frobenius norm is moved to the right during the orthogonalization step, the transformations for the updated core and the next core to be updated become

$$(\mathbf{R}_n^{\mathrm{L}})^{-\top} \otimes \mathbf{I} \quad \text{and} \quad \mathbf{I} \otimes \mathbf{R}_n^{\mathrm{L}}, \tag{4.9}$$

respectively. It can be easily shown that the transformations for the orthogonalization step do not affect the statistical independence of the joint distribution of the random variables, since the transformations are performed on each variable individually. The following example shows the updating for the transformation scheme of the random variables that represent an exemplary three core TT.

Example 4.1 (distribution updates and orthogonalization transformations for a TT with three cores). Assume we would like to apply Theorem 3.3 to find a TT with three cores and keep the TD in site-n-mixed-canonical form. The three TT-cores are initialized with a prior distribution and transformed such that the corresponding TT is in site-1-mixed-canonical form. The random variables that represent the orthogonal cores are denoted by $\mathbf{q}_i,\ i = 1, 2, 3$, and the random variable representing the TT-core that contains the Frobenius norm is denoted by $\mathbf{x}_i,\ i = 1, 2, 3$. After the random variables are transformed into site-1-mixed-canonical form using (4.7) and (4.8), the first core is updated followed by moving the Frobenius norm to the second core. Then the second core is updated and the Frobenius norm is moved to the last. When this half-sweep, as shown below, is completed using the transformations from (4.9), the same procedure is repeated in the opposite direction, requiring again (4.7) and (4.8). The example is depicted in Figure 12.

The ALS in a Bayesian framework with orthogonalization step has another difference compared to the one without orthogonalization. The update equations for the mean and covariance, (3.3) and (3.4), are affected by the TT decomposition being in site-n-mixed-canonical form: the matrix $\mathbf{U}_{\setminus n}$ becomes orthogonal and the update equations simplify to

$$\mathbf{m}_n^+ = \underbrace{\left[(\mathbf{P}_n^0)^{-1} + \frac{\mathbf{I}}{\sigma^2}\right]^{-1}}_{\mathbf{P}_n^+} \left[\frac{\mathbf{U}_{\setminus n}^\top \mathbf{y}}{\sigma^2} + (\mathbf{P}_n^0)^{-1}\mathbf{m}_n^0\right].$$

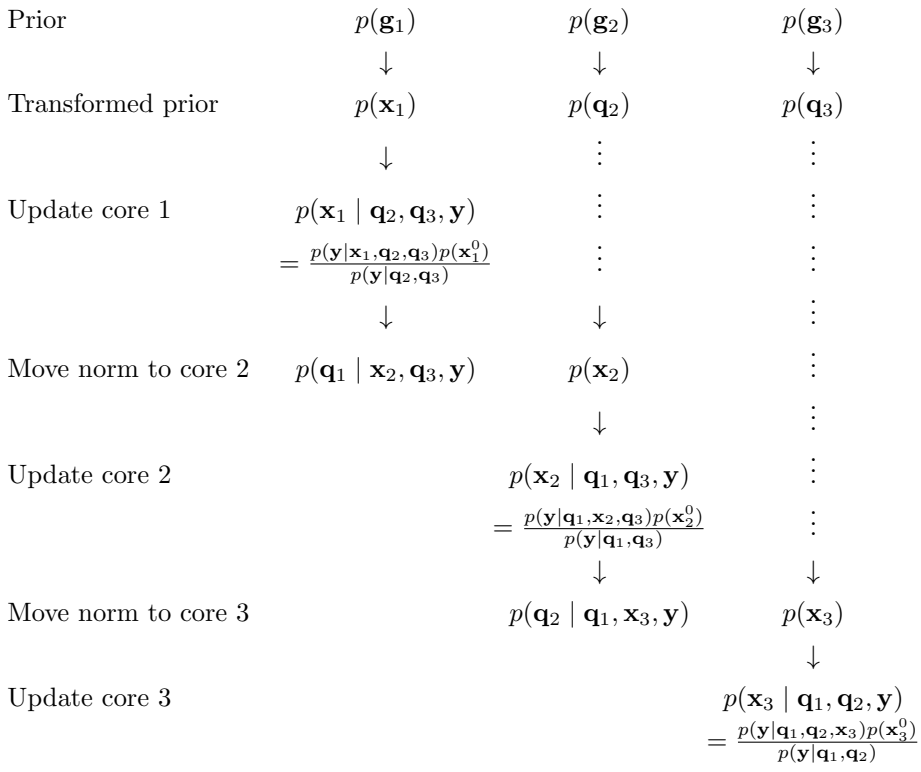

FIG. 12. *Distribution updates with orthogonalization step for example with three TT-cores.*

In this case $\mathbf{U}_{\setminus n}^\top \mathbf{y}$ corresponds to the update of the conventional ALS (3.6), due to the orthogonality of $\mathbf{U}_{\setminus n}$.

Algorithm 4.1 summarizes the ALS in a Bayesian framework with orthogonalization step for a TT decomposition. The computational cost of one update in Algorithm 4.1 is $\mathcal{O}(R^6 I^3)$ for the inversion and $\mathcal{O}(R^3 I)$ for the thin **QR**-factorization, and the storage requirement is $\mathcal{O}(R^2 I + R^4 I^2)$, where R is the largest TT-rank and I is the largest dimension of the approximated tensor. The only difference compared to the conventional ALS in terms of complexity is the additionally required storage for the covariance matrices. Thus, the number of elements of one TD component, depending on the ranks, will be the limiting factor for the computational complexity.

Algorithm 4.1. ALS in Bayesian framework with orthogonalization step.

Require: Prior mean $\{\mathbf{m}_i^0\}$ and covariance $\{\mathbf{P}_i^0\}$, $i = 1, \dots, N$, measurement $\mathbf{y}$ and noise variance σ^2.

Ensure: Posterior mean $\{\mathbf{m}_i^+\}$ and covariance $\{\mathbf{P}_i^+\}$, $i = 1, \dots, N$.

1: Transform random variables such that the corresponding TT decomposition is in site-1-mixed-canonical form.

2: Set $\{\mathbf{m}_i\} := \{\mathbf{m}_i^0\}$, $\{\mathbf{P}_i\} := \{\mathbf{P}_i^0\}$, $i = 1, \dots, N$.

3: **while** stopping criterion is not true **do**

4: **for** $n = 1, \dots, N, N-1, \dots, 2$ **do**

5: Compute $\mathbf{U}_{\setminus n}$ with (2.3) for TT using the mean of the TD components $\{\mathbf{m}_i\}_{i \neq n}$.

6: $\mathbf{P}_n^+ \leftarrow \left[(\mathbf{P}_n^0)^{-1} + \frac{\mathbf{I}}{\sigma^2}\right]^{-1}$

7: $\mathbf{m}_n^+ \leftarrow \mathbf{P}_n^+ \left[\frac{\mathbf{U}_{\setminus n}^\top \mathbf{y}}{\sigma^2} + \left(\mathbf{P}_n^0\right)^{-1} \mathbf{m}_n^0\right]$

8: **if** next core is to the right, **then**

9: $\mathbf{m}_n^+ \leftarrow \mathrm{vec}(\mathbf{Q}_n^{\mathrm{L}})$, with $\mathbf{Q}_n^{\mathrm{L}}$ from thin $\mathbf{QR}$-factorization of $\mathbf{G}_n^{\mathrm{L}}$

10: $\mathbf{P}_n^+ \leftarrow \left(\left(\mathbf{R}_n^{\mathrm{L}}\right)^{-\top} \otimes \mathbf{I}\right) \mathbf{P}_n^+ \left(\left(\mathbf{R}_n^{\mathrm{L}}\right)^{-1} \otimes \mathbf{I}\right)$

11: $\mathbf{m}_{n+1} \leftarrow (\mathbf{I} \otimes \mathbf{R}_n^{\mathrm{L}}) \mathbf{m}_{n+1}$

12: $\mathbf{P}_{n+1} \leftarrow (\mathbf{I} \otimes \mathbf{R}_n^{\mathrm{L}}) \, \mathbf{P}_{n+1} \, (\mathbf{I} \otimes \left(\mathbf{R}_n^{\mathrm{L}}\right)^\top)$

13: **else if** next core is on the left, **then**

14: $\mathbf{m}_n^+ \leftarrow \mathrm{vec}\left(\left(\mathbf{Q}_n^{\mathrm{R}}\right)^\top\right)$, with $\mathbf{Q}_n^{\mathrm{R}}$ from thin $\mathbf{QR}$-factorization of $\mathbf{G}_n^{\mathrm{R}}$

15: $\mathbf{P}_n^+ \leftarrow \left(\mathbf{I} \otimes \left(\mathbf{R}_n^{\mathrm{R}}\right)^{-\top}\right) \mathbf{P}_n^+ \left(\mathbf{I} \otimes \left(\mathbf{R}_n^{\mathrm{R}}\right)^{-1}\right)$

16: $\mathbf{m}_{n-1} \leftarrow (\mathbf{R}_n^{\mathrm{R}} \otimes \mathbf{I}) \mathbf{m}_{n-1}$

17: $\mathbf{P}_{n-1} \leftarrow (\mathbf{R}_n^{\mathrm{R}} \otimes \mathbf{I}) \, \mathbf{P}_{n-1} \left(\left(\mathbf{R}_n^{\mathrm{R}}\right)^\top \otimes \mathbf{I}\right)$

18: **end if**

19: $\mathbf{m}_n \leftarrow \mathbf{m}_n^+$, $\mathbf{P}_n \leftarrow \mathbf{P}_n^+$

20: Apply the transformations of the lines 7–10 or 12–15 to $\mathbf{m}_n^0, \mathbf{P}_n^0$.

21: **end for**

22: **end while**

5. Unscented transform in TT format. In Algorithms 3.1 and 4.1 we compute the posterior distributions of the TT-cores $p(\mathbf{g}_n \mid \{\mathbf{g}_i\}_{i \neq n}, \mathbf{y})$. However, we are interested in computing the distribution of the low-rank tensor estimate $\boldsymbol{\mathcal{G}}$, which is computed with a nonlinear function dependent on the posterior distributions and is, therefore, not Gaussian. The unscented transform (UT) [22] can approximate the mean $\mathbf{m}_{\mathrm{UT}}$ and covariance $\mathbf{P}_{\mathrm{UT}}$ of the sought distribution. In this section, we show how we can perform the UT in TT format. In this way, the direct computation of the potentially large covariance matrix can be avoided.

Generally, the UT approximates the mean and covariance of a distribution that is a nonlinear function of a known distribution $\mathbf{h} \sim \mathcal{N}(\mathbf{m}, \mathbf{P})$ with mean $\mathbf{m} \in \mathbb{R}^{M\times 1}$ and covariance $\mathbf{P} \in \mathbb{R}^{M\times M}$ [32, pp. 81–84]. First, $2M+1$ sigma points are formed with

$$\mathbf{x}^{(0)} = \mathbf{m}, \tag{5.1}$$

$$\mathbf{x}^{(i)} = \mathbf{m} + \sqrt{M+\lambda}\ \left[\sqrt{\mathbf{P}}\right]_i, \quad i = 1, \dots, M, \tag{5.2}$$

$$\mathbf{x}^{(i+M)} = \mathbf{m} - \sqrt{M+\lambda}\ \left[\sqrt{\mathbf{P}}\right]_i, \quad i = 1, \dots, M, \tag{5.3}$$

where the square root of the covariance matrix $\sqrt{\mathbf{P}}$ corresponds to the Cholesky factor, such that $\sqrt{\mathbf{P}}\sqrt{\mathbf{P}}^\top = \mathbf{P}$, where $\left[\sqrt{\mathbf{P}}\right]_i$ is the ith column of that matrix. The scaling parameter λ is defined as $\lambda = \alpha^2(M+\kappa) - M$, where α and κ determine the spread of the sigma points around the mean. Second, the sigma points are propagated through the nonlinearity. Third, the approximated mean $\mathbf{m}_{\mathrm{UT}}$ and covariance $\mathbf{P}_{\mathrm{UT}}$ are computed as

$$\mathbf{m}_{\mathrm{UT}} = \sum_{i=0}^{2M} w_i^{\mathbf{m}} \boldsymbol{\mathcal{S}}^{(i)}, \tag{5.4}$$

$$\mathbf{P}_{\mathrm{UT}} = \sum_{i=0}^{2M} w_i^{\mathbf{P}} \left(\boldsymbol{\mathcal{S}}^{(i)} - \mathbf{m}_{\mathrm{UT}}\right)\left(\boldsymbol{\mathcal{S}}^{(i)} - \mathbf{m}_{\mathrm{UT}}\right)^\top, \tag{5.5}$$

where $\boldsymbol{\mathcal{S}}^{(i)}$ are the transformed sigma points. The scalars $w_i^{\mathbf{m}}$ and $w_i^{\mathbf{P}}$ denote weighting factors, defined as

$$w_0^{\mathbf{m}} = \frac{\lambda}{M+\lambda}, \quad w_0^{\mathbf{P}} = w_0^{\mathbf{m}} + \left(1 - \alpha^2 + \beta\right),$$
$$w_i^{\mathbf{m}} = w^{\mathbf{m}} = w_i^{\mathbf{P}} = w^{\mathbf{P}} = \frac{1}{2(M+\lambda)}, \quad i = 1, \dots, 2M.$$

Literature suggests $\alpha = 0.001$, $\kappa = 3 - M$ [16, p. 229], and for Gaussian distributions $\beta = 2$ [32, p. 229].

In order to use the UT in TT format, the known distribution $\mathbf{h} \sim \mathcal{N}(\mathbf{m}, \mathbf{P})$ is computed from the cores' mean and covariance as

$$\mathbf{h} \sim \mathcal{N}(\mathbf{m}, \mathbf{P}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{m}_1\\ \mathbf{m}_2\\ \vdots\\ \mathbf{m}_N\end{bmatrix}, \begin{bmatrix}\mathbf{P}_1 & 0 & \dots & 0\\ 0 & \mathbf{P}_2 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \dots & 0 & \mathbf{P}_N\end{bmatrix}\right). \tag{5.6}$$

The mean, consisting of the stacked vectorized cores, is of size $M \times 1$ with

$$M = \sum_{n=1}^{N} R_n I_n R_{n+1}, \quad R_1 = R_{N+1} = 1.$$

The covariance matrix of size $M \times M$ is block diagonal, since we assume the TT-cores to be statistically independent. The nonlinear function for the UT in TT format is defined as

$$f_{\mathrm{T}} : \mathbb{R}^{M\times 1} \to \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N} \text{ given by } \mathbf{x} \mapsto \boldsymbol{\mathcal{G}}, \tag{5.7}$$

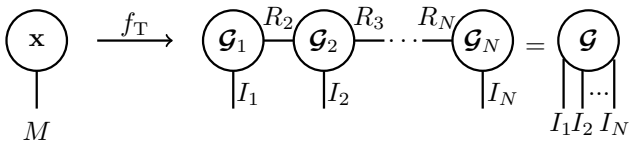

FIG. 13. *Visual depiction of the nonlinear transformation from vector* $\mathbf{x}$ *into a TT with cores* $\boldsymbol{\mathcal{G}}_i$, $i = 1, \ldots, N$, *that represents tensor* $\boldsymbol{\mathcal{G}}$.

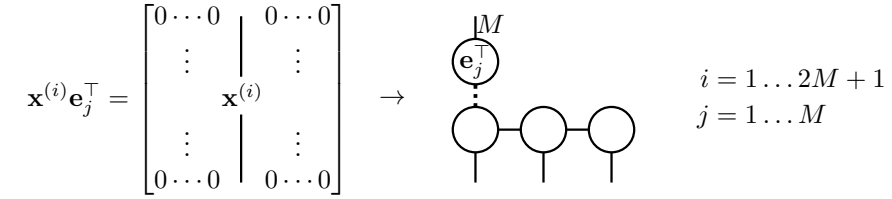

FIG. 14. *Propagation of each sigma point as a column of matrix* $\mathbf{x}^{(i)}\mathbf{e}_i^\top$ *through the nonlinearity.*

where $\mathbf{x}$ is a vector of size $M \times 1$. The transformation of a vector into a tensor is depicted in Figure 13.

The formation and propagation of the sigma points works as follows. The first sigma point $\mathbf{x}^{(0)}$ is the mean $\mathbf{m}$ from (5.6) and propagated through the nonlinearity; it corresponds to the TT represented by the distributions determined by Algorithm 3.1. To facilitate later steps, the remaining sigma points from (5.2) and (5.3) are organized into two matrices, according to

$$\mathbf{A}_+ = \sum_{i=1}^{M} \mathbf{x}^{(i)}\mathbf{e}_i^\top, \tag{5.8}$$

$$\mathbf{A}_- = \sum_{i=1}^{M} \mathbf{x}^{(M+i)}\mathbf{e}_i^\top, \tag{5.9}$$

where $\mathbf{e}_i$ denotes a vector with zeros everywhere except a 1 at location i. In this way, the propagation through the nonlinearity of all sigma points then becomes a propagation of every summand $\mathbf{x}^{(i)}\mathbf{e}_i^\top$ and $\mathbf{x}^{(M+i)}\mathbf{e}_i^\top$, respectively. The propagation is achieved by forming TT-cores from $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(M+i)}$ and summing over a rank-1 connection between the first core and the vector $\mathbf{e}_i^\top$ as shown in Figure 14.

Then, all summands of $\mathbf{A}_+ \in \mathbb{R}^{M \times M}$ and $\mathbf{A}_- \in \mathbb{R}^{M \times M}$, are summed together, respectively, by stacking the cores according to [28, p. 2308], as illustrated in Figure 15 (left). The summation causes the ranks of the TTm to increase and a rounding procedure [27, pp. 2301–2305] needs to be applied to reduce the ranks back to the required precision. Finally, the vector containing the weights $\mathbf{w}^{\mathbf{m}} = w^{\mathbf{m}}\mathbf{1}_M$ is absorbed into the first core (Figure 15, right).

The computation of the covariance matrix from (5.5) is divided into two steps. First, $\mathbf{m}_{\mathrm{UT}}$ is subtracted from the sum over the sigma points (Figure 15, left). This is achieved by creating a matrix, where $\mathbf{m}_{\mathrm{UT}}$ is stacked M times next to each other. The visual depiction of this operation equals the one in Figure 14 with the difference that the multiplied vector is $\mathbf{1}_M^\top$. Second, the result from the first step absorbs the weights into the first core, as depicted in Figure 16. The resulting covariance matrix

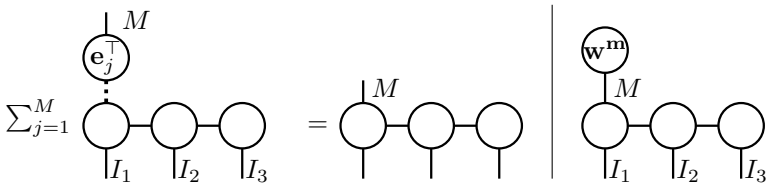

FIG. 15. *Visual depiction of* (5.4) *as a sum over sigma points (left) and absorption of the weight vector* $\mathbf{w}^\mathbf{m}$ *into the first core (right).*

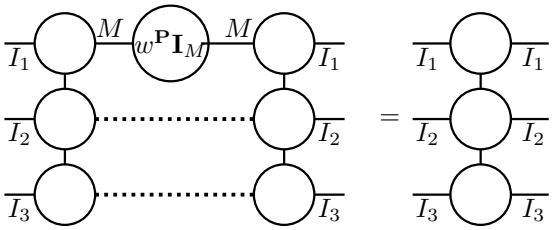

FIG. 16. *Visual depiction of* (5.5) *with* $w^\mathbf{P}\mathbf{I}_M$ *as a diagonal matrix containing the weight factors on the diagonal.*

$\mathbf{P}_{\mathrm{UT}}$ in the three-core example is a TT matrix that corresponds to a matrix of size $I_1I_2I_3 \times I_1I_2I_3$.

The computation of the approximate mean and covariance with the UT in TT format is summarized in Algorithm 5.1. The computational cost depends on the ranks of the TD, since M is a function of the ranks. The bottleneck of Algorithm 5.1 is the rounding procedure necessary after performing summations in TT format. It has a cost of $\mathcal{O}(R^3I^2N)$ for a TT matrix.

Algorithm 5.1. Approximation of the low-rank tensor estimate's mean and covariance with the unscented transform in TT format.

Require: The mean and covariances of each TT-core $\{\mathbf{m}_i, \mathbf{P}_i\}$, $i = 1, \ldots, N$, computed with the ALS in a Bayesian framework.

Ensure: The approximated mean $\mathbf{m}_{\mathrm{UT}}$ and covariance $\mathbf{P}_{\mathrm{UT}}$ in TT format of the low-rank tensor estimate's distribution.

1: Compute sigma point $\mathbf{x}^{(0)}$ with (5.1).
2: Compute remaining sigma points with (5.2) and (5.3) and organize them into groups according to (5.8) and (5.9).
3: Propagate sigma points through (5.7), where groups from step 2 are propagated as shown in Figure 13.
4: Estimate the mean $\mathbf{m}_{\mathrm{UT}}$ with (5.4) as shown in Figure 15.
5: Estimate the covariance $\mathbf{P}_{\mathrm{UT}}$ with (5.5) as shown in Figure 16.

6. Numerical experiments. In this section, we present the numerical examples that test the algorithms. All experiments with exception of the last were performed with MATLAB R2020b on a Dell computer with processor Intel Core i7-8650U CPU @ 1.90GHz 2.11 GHz and 8GB of RAM. The last experiment is performed on a Lenovo computer with processor Intel Core i7-10700KF CPU @ 3.80GHz 3.79 GHz and 16GB of RAM. The implementation of the experiments can be found on https:

//gitlab.tudelft.nl/cmmenzen/bayesian-als.

The first three experiments are performed with a random TT, $\boldsymbol{\mathcal{G}}$, that represents the ground truth and has the cores

$$\boldsymbol{\mathcal{G}}_{1,\text{truth}} \in \mathbb{R}^{1\times 5\times 3}, \quad \boldsymbol{\mathcal{G}}_{2,\text{truth}} \in \mathbb{R}^{3\times 5\times 3}, \text{ and } \boldsymbol{\mathcal{G}}_{3,\text{truth}} \in \mathbb{R}^{3\times 5\times 1}.$$

The entries of each TT-core are drawn from a standard normal distribution. After computing the tensor $\boldsymbol{\mathcal{Y}}_{\text{truth}} \in \mathbb{R}^{5\times 5\times 5}$ from the TT-cores and vectorizing it, a noisy sample $\mathbf{y}$ is formed with

$$\mathbf{y} = \mathbf{y}_{\text{truth}} + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}), \tag{6.1}$$

where $\mathbf{y}_{\text{truth}}$ denotes the vectorized ground truth and $\boldsymbol{\epsilon}$ is a realization of random noise. The noisy samples of the same underlying tensor formed with (6.1) are uncorrelated. The covariance of the measurement noise is influenced by fixing the signal-to-noise ratio

$$\text{SNR}_{\text{dB}} = 10\log_{10}\frac{\|\mathbf{y}\|^2}{\|\boldsymbol{\epsilon}\|^2}.$$

If not stated otherwise, the signal-to-noise ratio is set to zero dB. Some experiments use multiple noisy samples $\mathbf{y}$, computed from (6.1). In this case, the estimate is recursively updated. Initially, the prior TT is input to Algorithm 3.1 together with a sample $\mathbf{y}$. After the execution of Algorithm 3.1, the output mean and covariance is used as a prior for the next execution together with a new sample $\mathbf{y}$. This recursive updating is very suitable for the ALS in a Bayesian framework, because it can deal with prior knowledge on the TD components. For the conventional ALS, the estimate from an execution of the algorithm that computes a TT with the ALS is used as an initial TT for the next execution.

6.1. Convergence analysis of maximization problem. In the ALS in a Bayesian framework, we solve the optimization problem given by (1.2). In this context, we define the relative error between the low-rank estimate $\mathbf{g}$ and the ground truth $\mathbf{y}_{\text{truth}}$ as

$$\varepsilon_{\text{truth}} = \frac{\|\mathbf{y}_{\text{truth}} - \mathbf{g}\|}{\|\mathbf{y}_{\text{truth}}\|},$$

and the relative error between the low-rank estimate $\mathbf{g}$ and the noisy sample $\mathbf{y}$ as

$$\varepsilon_{\text{meas}} = \frac{\|\mathbf{y} - \mathbf{g}\|}{\|\mathbf{y}\|}.$$

In the first experiment, we look at the errors defined above in order to analyze the convergence of Algorithm 3.1. In addition, we look at the evolution of the log likelihood times the prior, since from Theorem 3.3 it follows that the numerator of the logarithm of (1.4) needs to be maximized to compute the posterior of all TD components.

In this experiment, only one noisy sample $\mathbf{y}$ is used. The prior mean is initialized randomly and the covariance for each core is set to $200^2\mathbf{I}$, meaning a low certainty on the prior mean. The experiment is performed 100 times with the same TT, $\boldsymbol{\mathcal{G}}$, but with different priors. Then, the mean of the 100 results is plotted with a region of twice the standard deviation. Figure 17, left shows how both relative errors decrease rapidly and converge after approximately five iterations in Algorithm 3.1. Figure 17, right shows how the product of log likelihood and prior increases during the first approximately six iterations, converging to a fixed value. Both subfigures of Figure 17 also show how the region of twice the standard deviation from the 100 trials becomes smaller with an increasing number of iterations. Hence, it can be concluded that Algorithm 3.1 converges and therefore also the optimization problem converges.

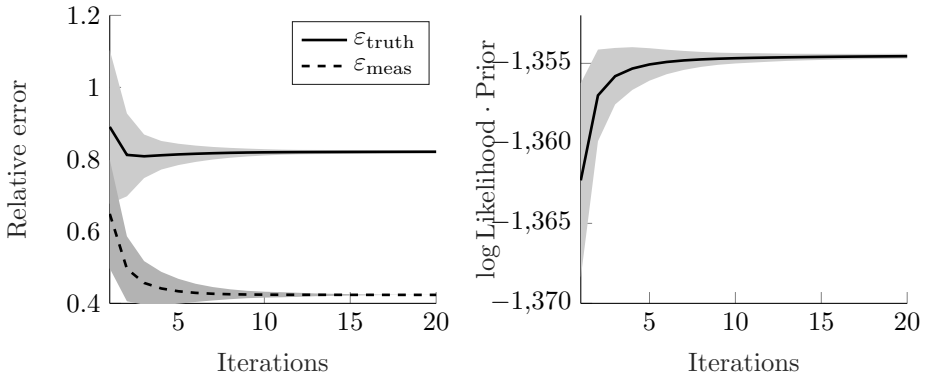

FIG. 17. *Left: Evolution of the relative errors during* 20 *iterations in Algorithm* 3.1. *Right: Evolution of log likelihood times the prior during* 20 *iterations in Algorithm* 3.1.

6.2. Analysis of covariance matrices. In the second experiment, we look at how the covariance matrix of each core changes throughout the iterations in Algorithm 4.1. We also examine the covariance matrix of the low-rank tensor estimate, computed with the UT in TT format. The experiment is performed 100 times with the same TT, $\boldsymbol{\mathcal{G}}$, but with different priors, as in subsection 6.1. Then, the mean of the 100 results is plotted with a region of twice the standard deviation. Figure 18 shows the trace and Frobenius norm of the covariance matrix of the core that will be updated next, after the norm is moved to this core. Both the trace and Frobenius norm of each core's covariance matrix decrease and converge to a fixed value. For the first and third core, the values are smaller than for the second, because the second core has a larger number of elements. The convergence behavior is also shown in Figure 19, where the trace and Frobenius norm of the covariance matrix of the low-rank tensor estimate converge quickly to a fixed value. The decreasing and converging values of the trace (Figure 17, top) and the Frobenius norm (Figure 17, bottom) indicate that the uncertainty of the mean decreases and then remains constant with an increasing number of iterations. In the next experiments, we will use the information of the covariance matrices to visualize a confidence interval for our estimate.

6.3. Comparison to conventional ALS. The main benefits of the ALS in a Bayesian framework are the uncertainty quantification of the low-rank tensor estimate, as well as the incorporation of prior knowledge. In the third experiment, we show the benefits by comparing the ALS in a Bayesian framework to the conventional ALS.

Figure 20 depicts the vectorized low-rank tensor estimate for the ALS and the mean of the ALS in a Bayesian framework's estimate with a 95% confidence interval in comparison with the ground truth. The uncertainty measure is computed from the diagonal elements of $\mathbf{P}_{\text{UT}}$. The top figure shows the estimate using one noisy sample $\mathbf{y}$ for the ALS in a Bayesian framework and the bottom using 100 noisy samples. While the ALS does not improve when taking into account multiple noisy samples, the ALS in a Bayesian framework improves in two aspects. The error between the mean and the truth becomes smaller and the estimate becomes more certain. Also, in the top figure, where the uncertainty is relatively large, the ground truth almost always lies inside the confidence interval and therefore the ALS in a Bayesian framework provides more information than the conventional ALS.

Now, we analyze the influence of the prior quality on the relative error. Figure 21

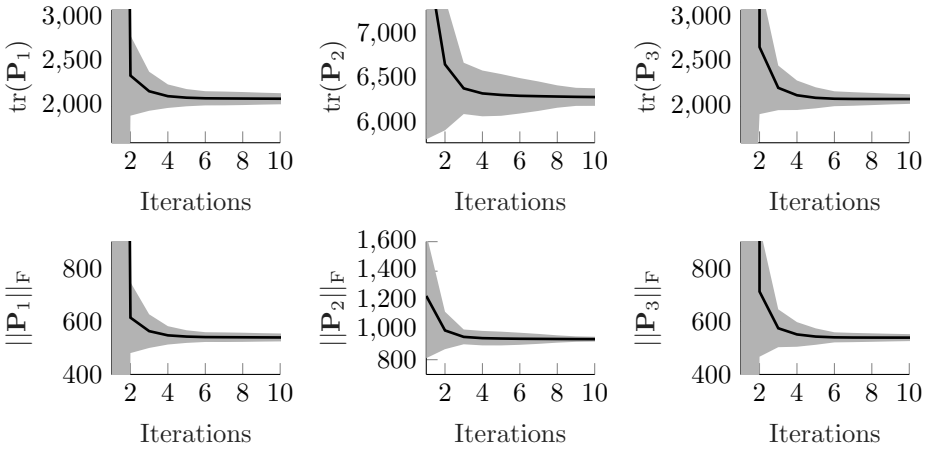

FIG. 18. *Top: Trace. Bottom: Frobenius norm of covariance matrix of* $\boldsymbol{\mathcal{G}}_1$, $\boldsymbol{\mathcal{G}}_2$, *and* $\boldsymbol{\mathcal{G}}_3$.

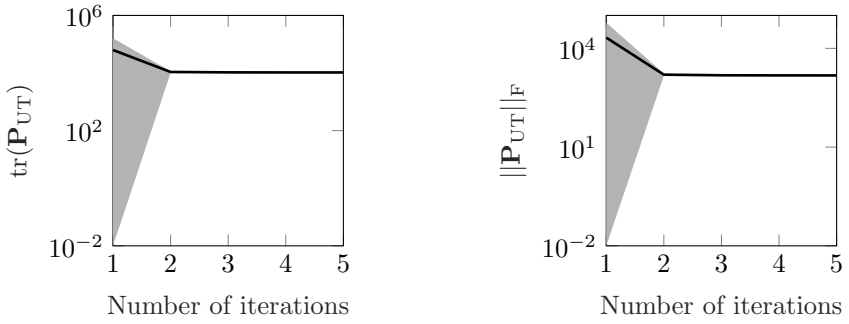

FIG. 19. *Left: Trace. Right: Frobenius norm of covariance matrix of low-rank tensor estimate.*

shows the relative error $\varepsilon_{\text{truth}}$ of the ALS in a Bayesian framework for different priors. The prior mean is computed from

$$\mathbf{m}_i^0 = \mathbf{g}_{i,\text{truth}} + a\,\mathcal{N}(\mathbf{0},\mathbf{I}), \quad i = 1,2,3, \tag{6.2}$$

where $\mathbf{g}_{i,\text{truth}}$ denotes the vectorization of $\boldsymbol{\mathcal{G}}_{i,\text{truth}}$ and a is a number that is set to values between 0 and 5. It determines how different the prior mean is from the ground truth. The prior covariance is computed from

$$\mathbf{P}_i^0 = b^2\,\mathbf{I}, \quad i = 1,2,3, \tag{6.3}$$

by setting b to values between 0 and 5. A small value means a high certainty and a large value means a low certainty on the prior mean. Figure 21 shows that the error is small if the prior mean is close to the ground truth and the covariance is small. For a bad prior and a small covariance, the error is 100 percent or larger, since a high certainty for a bad prior is assumed. For comparison, the isoline (dashed line) corresponding to the mean relative error of the conventional ALS is shown in the graph, which is almost independent of the prior information.

Figure 22, left shows the relative error of the reconstructed tensor versus the signal-to-noise ratio for a prior mean from (6.2) with $a = 10^{-1}$ and prior covariance

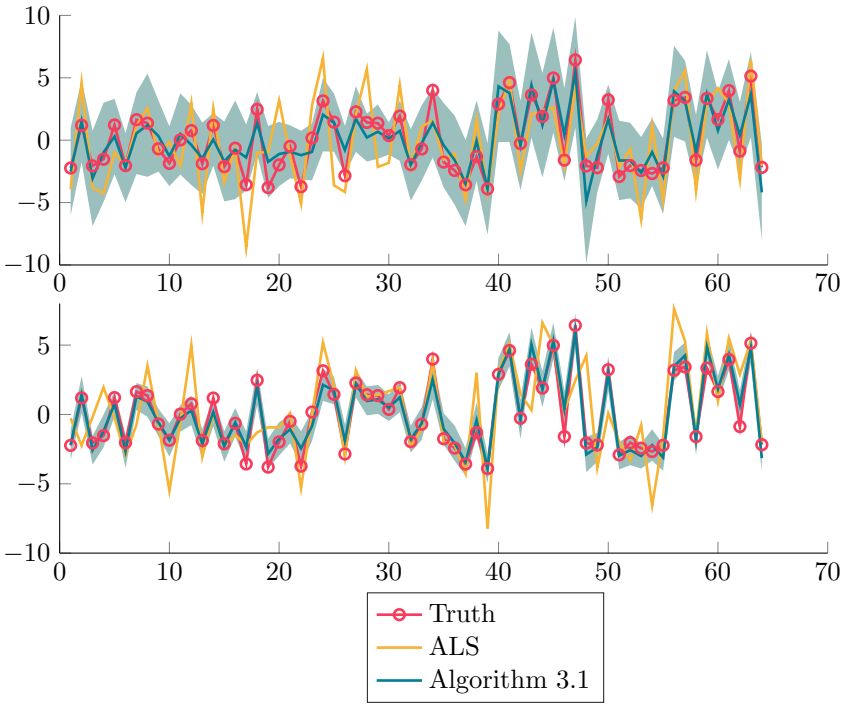

FIG. 20. *Ground truth with ALS estimate and mean of estimate from the ALS in a Bayesian framework with confidence region of* 95%. *Top: Estimate using one noisy sample. Bottom: Estimate using* 100 *noisy samples.*

from (6.3) with $b = 10^{-1}$, meaning a good prior and a high certainty on the prior. While the ALS performs poorly for high noise, the ALS in a Bayesian framework results in small relative errors. For an increasing SNR, the relative error of the ALS in a Bayesian framework converges to the one of the ALS.

Further, Figure 22, right shows the ALS in comparison with the ALS in a Bayesian framework for multiple noisy samples. While the relative error $\varepsilon_{\text{truth}}$ decreases for the ALS in a Bayesian framework, the conventional ALS does not improve when more noisy samples become available.

As shown, the ALS in a Bayesian framework gives better results if a good prior is available and it provides a measurement of the uncertainty and, therefore, additional valuable information. Also, if multiple noisy samples are available, ALS in a Bayesian framework significantly improves the estimate.

6.4. Reconstruction of noisy image. To test Algorithm 3.1 on an image processing problem, a cat image is reconstructed from an image corrupted with noise. Figure 23 shows the steps before applying Algorithm 3.1. The original image of size 256×256 pixel is reshaped into an 8-way tensor, where each mode is of dimension 4. To obtain the TT-ranks, here we use the TT-SVD algorithm [27]. It finds a TD that approximates the given tensor by setting an upper bound for the relative error. With an upper bound of 0.1, the TT-ranks, depicted in Figure 23, are obtained. Finally, the ground truth is computed as the vectorized contracted TT. Now, ten noisy samples are formed with (6.1) with a signal-to-noise ratio of $\text{SNR}_{\text{dB}} = 0$.

Figure 24 (a) shows the original image and Figure 24 (b) the low-rank image,

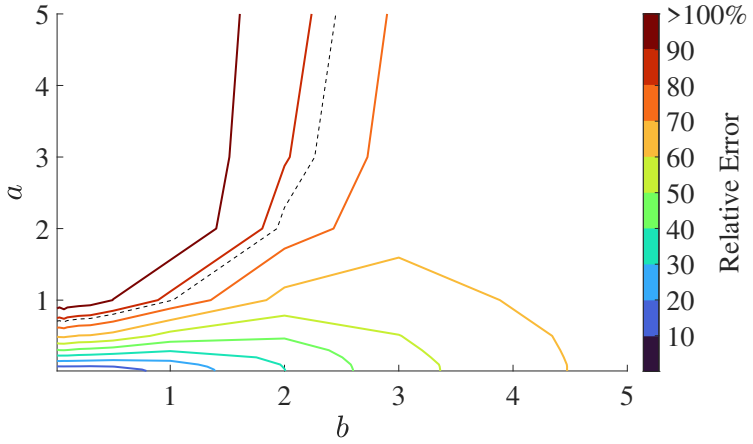

FIG. 21. *Relative error of the ALS in a Bayesian framework for different priors for* $\mathrm{SNR}_{\mathrm{dB}} = 0$. *The y-axis indicates the similarity of the prior mean to the ground truth and the x-axis indicates the certainty on the prior mean. The dashed line corresponds to the isoline corresponding to the mean error of the conventional ALS.*

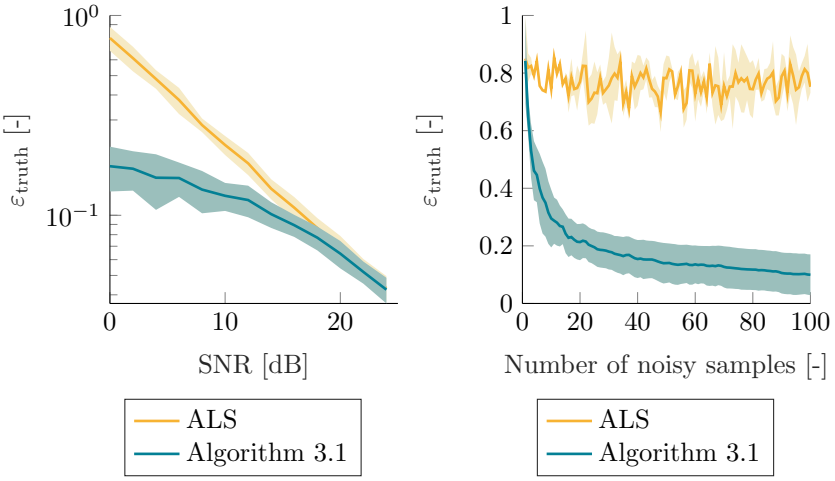

FIG. 22. *Left: Relative error* $\varepsilon_{\mathrm{truth}}$ *vs. signal-to-noise ratio with prior mean from* (6.2) *with* $a = 10^{-1}$ *and prior covariance from* (6.3) *with to* $b = 10^{-1}$. *Right: Comparison of the the relative error* $\varepsilon_{\mathrm{truth}}$ *between the ALS and the ALS in a Bayesian framework for different numbers of noisy samples.*

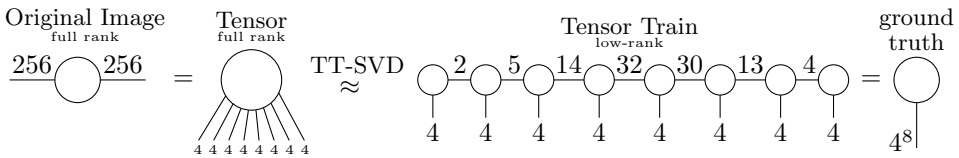

FIG. 23. *Computation of ground truth from the original image. The original image of size* 256×256 *pixel is reshaped into an* 8*-way tensor, where each mode is of dimension* 4. *Then, the TT-SVD algorithm* [27] *with an upper bound for the relative error of* 0.1 *is applied, resulting in the depicted TT-ranks. Finally, the ground truth is obtained as the vectorized contracted TT.*

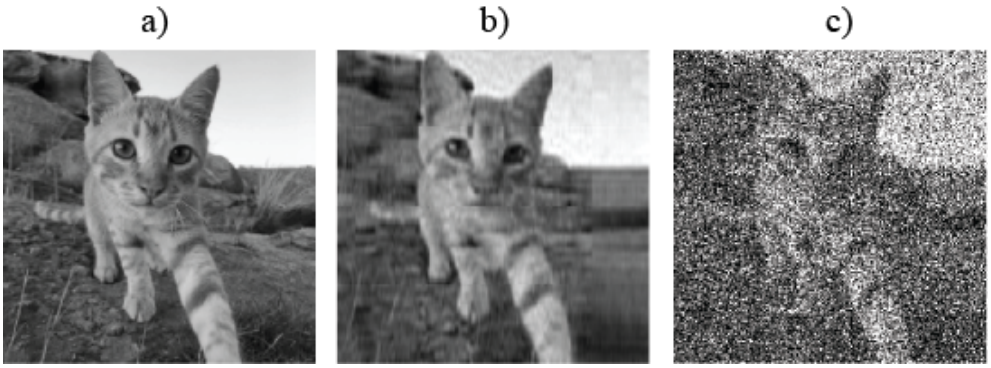

FIG. 24. (a) *Original image.* (b) *Image approximated with the TT-SVD algorithm* [27] *with an upper bound for the relative error of* 0.1. (c) *One noisy sample (low-rank image corrupted with random noise).*

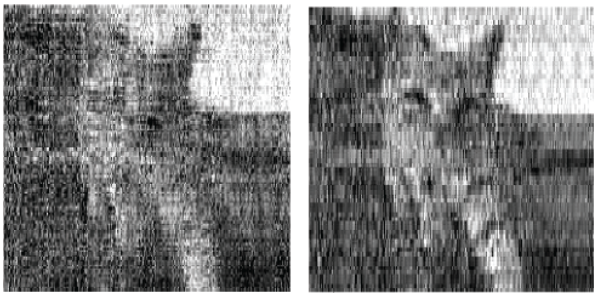

FIG. 25. *Reconstructed image with conventional ALS algorithm. Left: using one noisy sample. Right: using ten noisy samples.*

which is obtained by reshaping the low-rank TT from the TT-SVD into the size of the original image. Figure 24 (c) shows one exemplary noisy sample $\mathbf{y}$ reshaped into the dimensions of the original image. As a stopping criterion, we used the maximum number of iterations of 3. Figure 25, left shows the reconstruction of the image with the conventional ALS using one noisy sample and on the right using ten noisy samples. Figure 26 shows the reconstruction of the image inputting a random prior mean and a prior covariance on each core of $1000^2\mathbf{I}$ and using one and ten noisy samples. For the ALS in a Bayesian framework, it is shown that the image gets clearer with a higher number of noisy samples $\mathbf{y}$, confirmed by the decreasing relative error $\varepsilon_{\text{truth}}$ from 0.3127 to 0.1478. The relative error of the conventional ALS only decreases slightly from 0.3664 to 0.3088.

6.5. Large-scale experiment. In this experiment, we demonstrate that Algorithm 3.1 also works with larger tensors. The cat image from subsection 6.4 in color is upscaled via bicubic interpolation to obtain a $6000 \times 4000 \times 3$ tensor as depicted in Figure 27 (a). Next, we find a low-rank approximation of the image by first applying the TKPSVD algorithm [2]. The TKPSVD decomposes a tensor $\boldsymbol{\mathcal{A}}$ into a sum of multiple Kronecker products of N tensors $\boldsymbol{\mathcal{A}}_r^{(n)}$,

$$\boldsymbol{\mathcal{A}} = \sum_{r=1}^{R} \lambda_r \, \boldsymbol{\mathcal{A}}_r^{(N)} \otimes \cdots \otimes \boldsymbol{\mathcal{A}}_r^{(1)},$$

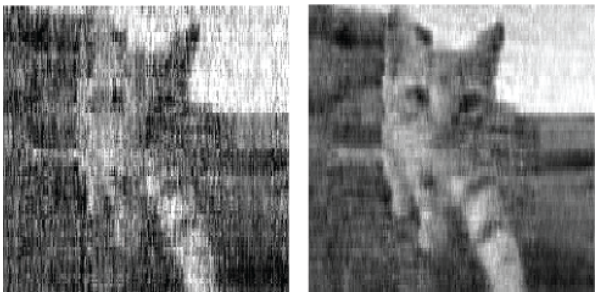

FIG. 26. *Reconstructed image with ALS in a Bayesian framework (Algorithm* 3.1*). Left: Using one noisy sample. Right: Using ten noisy samples.*

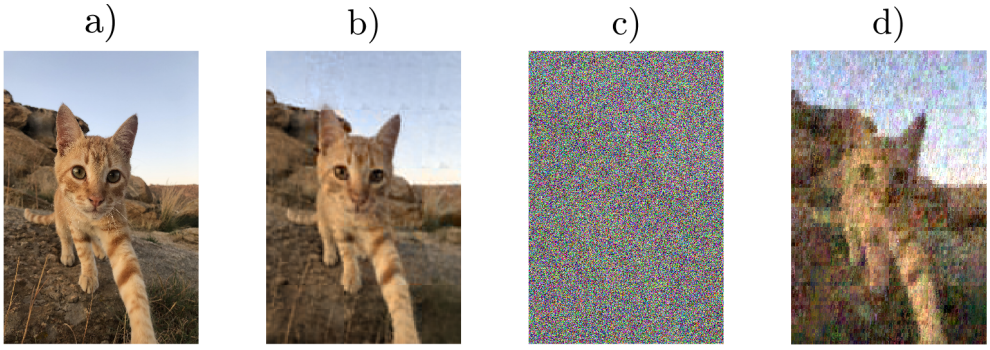

FIG. 27. (a) *Original image.* (b) *Low-rank image.* (c) *Noisy image (low-rank image corrupted with random noise, with an* SNR $= -22$*).* (d) *Reconstructed image.*

where $\lambda_r \in \mathbb{R}$. We approximate the image by taking only the term with the largest λ_r and $N = 5$,

$$\boldsymbol{\mathcal{A}} \approx \lambda_{\max} \boldsymbol{\mathcal{A}}_1^{(5)} \otimes \boldsymbol{\mathcal{A}}_1^{(4)} \otimes \boldsymbol{\mathcal{A}}_1^{(3)} \otimes \boldsymbol{\mathcal{A}}_1^{(2)} \otimes \boldsymbol{\mathcal{A}}_1^{(1)},$$

where $\lambda_{\max} = \lambda_1$. The resulting Kronecker product is of dimensions

$$(375 \times 250 \times 3) \otimes (2 \times 2 \times 1) \otimes (2 \times 2 \times 1) \otimes (2 \times 2 \times 1) \otimes (2 \times 2 \times 1),$$

as depicted in the top part of Figure 28. Second, $\boldsymbol{\mathcal{A}}_1^{(5)} \in \mathbb{R}^{375 \times 250 \times 3}$ is further decomposed with the TT-SVD algorithm with an upper bound of the relative error of 0.08, where the dimensions are factorized as shown in the lower part of Figure 28. The resulting low-rank approximation of the image is shown in Figure 27 (b) and the noisy image, created with an SNR $= -22$, is shown in Figure 27 (c). We use Algorithm 3.1 with a random prior mean and $\mathbf{P}_i^0 = 10000^2\mathbf{I}$. Figure 27 (d) shows the reconstructed image after 30 iterations in Algorithm 3.1. The main computational bottleneck is the inversion of the covariance matrix of the largest TD component (line 4 of Algorithm 3.1). Thus, the number of elements of a TD component, dependent on its ranks, is the limiting factor for the computational complexity. In this case, the largest TT-core has $13 \cdot 25 \cdot 18 = 5850$ elements; see the second-to-last TT-core in the lower part of Figure 28.

7. Conclusions. We approached the computation of low-rank TD from a Bayesian perspective. Assuming Gaussian priors for the TD components and Gaussian measurement noise and by applying a block coordinate descent, we were able to perform

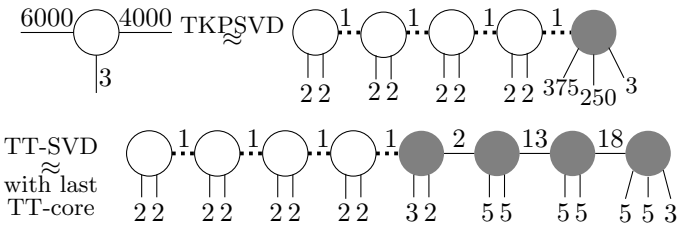

FIG. 28. *Determination of the TT-ranks by computing a low-rank decomposition with the TKPSVD and then decomposing the last TT-core (gray) further with the TT-SVD.*

a tractable inference and compute the posterior joint distribution of the TD components. This leads to a probabilistic interpretation of the ALS. The distribution of the underlying low-rank tensor was computed with the UT in TT format. We found that the relative error of the resulting low-rank tensor approximation depends strongly on the quality of the prior distribution. In addition, our method opens up for a recursive estimation of a tensor from a sequence of noisy measurements of the same underlying tensor. If no useful prior information is available, the method gives the same result as the conventional ALS. Our method will perform worse than the conventional ALS if a small covariance is assumed for a bad prior mean. Future work could focus on incorporating the inference of the ranks which for the ALS are fixed and therefore need to be decided beforehand. Also, the method could be extended to a non-Gaussian prior and the UT algorithm could be further developed, e.g., by parallelizing the code to make it computationally more efficient for large data sets.

REFERENCES

[1] K. BATSELIER, Z. CHEN, AND N. WONG, *Tensor network alternating linear scheme for MIMO Volterra system*, Automatica J. IFAC, 84 (2017), pp. 26–35.

[2] K. BATSELIER AND N. WONG, *A constructive arbitrary-degree Kronecker product*, Numer. Linear Algebra Appl., 24 (2017), e2097.

[3] C. F. CAIAFA AND A. CICHOCKI, *Stable, robust, and super fast reconstruction of tensors using multi-way projections*, IEEE Trans. Signal Process., 63 (2015), pp. 780–793.

[4] J. D. CARROLL AND J. J. CHANG, *Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition*, Psychometrika, 35 (1970), pp. 283–319.

[5] C. CHEN, K. BATSELIER, C.-Y. KO, AND N. WONG, *A support tensor train machine*, in Proceedings of the International Joint Conference on Neural Networks, IEEE, 2019, pp. 1–8.

[6] L. CHENG, Y.-C. WU, AND H. V. POOR, *Probabilistic tensor canonical polyadic decomposition with orthogonal factors*, IEEE Trans. Signal Process., 65 (2017), pp. 663–676.

[7] W. CHU AND Z. GHAHRAMANI, *Probabilistic models for incomplete multi-dimensional arrays*, J. Mach. Learn. Res., 5 (2009), pp. 89–96.

[8] A. CICHOCKI, N. LEE, I. V. OSELEDETS, A.-H. PHAN, Q. ZHAO, AND D. P. MANDIC, *Tensor networks for dimensionality reduction and large-scale optimization Part 1: Low-rank tensor decompositions*, Found. Trends Mach. Learn., 9 (2016), pp. 249–429.

[9] A. CICHOCKI, A.-H. PHAN, Q. ZHAO, N. LEE, I. V. OSELEDETS, M. SUGIYAMA, AND D. P. MANDIC, *Tensor networks for dimensionality reduction and large-scale optimization: Part 2 Applications and future perspectives*, Found. Trends Mach. Learn., 9 (2017), pp. 431–673.

[10] N. COHEN, O. SHARIR, AND A. SHASHUA, *On the expressive power of deep learning: A tensor analysis*, J. Mach. Learn. Res., 49 (2016), pp. 698–728.

[11] P. COMON, X. LUCIANI, AND A. DE ALMEIDA, *Tensor decompositions, alternating least squares and other tales*, J. Chemom., 23 (2009), pp. 393–405.

[12] S. V. DOLGOV AND D. SAVOSTYANOV, *Alternating minimal energy methods for linear systems in higher dimensions*, SIAM J. Sci. Comput., 36 (2014), pp. A2248–A2271, https://doi.org/10.1137/140953289.

[13] G. Favier, A. Y. Kibangou, and T. Bouilloc, *Nonlinear system modeling and identification using Volterra-PARAFAC models*, Internat. J. Adapt. Control Signal Process., 26 (2012), pp. 30–53.
[14] L. Grasedyck, M. Kluge, and S. Krämer, *Variants of alternating least squares tensor completion in the tensor train format*, SIAM J. Sci. Comput., 37 (2015), pp. A2424–A2450, https://doi.org/10.1137/130942401.
[15] R. Harshman, *Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis*, UCLA Working Papers in Phonetics, 16 (1970), pp. 1–84.
[16] S. S. Haykin, *Kalman Filtering and Neural Networks*, John Wiley & Sons, 2001.
[17] J. L. Hinrich, K. H. Madsen, and M. Mørup, *The probabilistic tensor decomposition toolbox*, Mach. Learn. Sci. Technol., 1 (2020), 025011.
[18] J. L. Hinrich and M. Mørup, *Probabilistic tensor train decomposition*, in Proceedings of the 27th European Signal Processing Conference, IEEE, 2019, pp. 1–5.
[19] P. D. Hoff, *Equivariant and scale-free Tucker decomposition models*, Bayesian Anal., 11 (2016), pp. 627–648.
[20] S. Holtz and T. Rohwedder, R. Schneider, *The alternating linear scheme for tensor optimization in the tensor train format*, SIAM J. Sci. Comput., 34 (2012), pp. A683–A713, https://doi.org/10.1137/100818893.
[21] P. A. Izmailov, A. V. Novikov, and D. A. Kropotov, *Scalable Gaussian processes with billions of inducing inputs via tensor train decomposition*, in Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, Proc. Mach. Learn. Res. 84, 2018, pp. 726–735.
[22] S. J. Julier and J. K. Uhlmann, *Unscented filtering and nonlinear estimation*, Proc. IEEE, 92 (2004), pp. 401–422.
[23] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500, https://doi.org/10.1137/07070111X.
[24] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., Springer, New York, 2006.
[25] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, *Tensorizing neural networks*, in Proceedings of the 28th International Conference on Neural Information Processing Systems, 2015, pp. 442–450.
[26] I. V. Oseledets, *Approximation of* $2d \times 2d$ *matrices using tensor decomposition*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2130–2145, https://doi.org/10.1137/090757861.
[27] I. V. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317, https://doi.org/10.1137/090752286.
[28] I. V. Oseledets and S. Dolgov, *Solution of linear systems and matrix inversion in the TT-format*, SIAM J. Sci. Comput., 34 (2012), pp. A2718–A2739, https://doi.org/10.1137/110833142.
[29] P. Rai, Y. Wang, and L. Carin, *Leveraging features and networks for probabilistic tensor decomposition*, in Proceedings of the National Conference on Artificial Intelligence, Vol. 4, 2015, pp. 2942–2948.
[30] P. Rai, Y. Wang, S. Guo, G. Chen, D. Dunson, and L. Carin, *Scalable Bayesian low-rank decomposition of incomplete multiway tensors*, in Proceedings of the 31st International Conference on Machine Learning, Vol. 5, 2014, pp. 3810–3820.
[31] T. Rohwedder and A. Uschmajew, *On local convergence of alternating schemes for optimization of convex problems in the tensor train format*, SIAM J. Numer. Anal., 51 (2013), pp. 1134–1162, https://doi.org/10.1137/110857520.
[32] S. Särkkä, *Bayesian Filtering and Smoothing*, Cambridge University Press, Cambridge, 2013.
[33] U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, Ann. Physics, 326 (2011), pp. 96–192.
[34] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, *Tensor decomposition for signal processing and machine learning*, IEEE Trans. Signal Process., 65 (2017), pp. 3551–3582.
[35] M. Signoretto, L. De Lathauwer, and J. A. Suykens, *A kernel-based framework to tensorial data analysis*, Neural Netw., 24 (2011), pp. 861–874.
[36] Q. Song, H. Ge, J. Caverlee, and X. Hu, *Tensor completion algorithms in big data analytics*, ACM Trans. Knowl. Discov. Data, 13 (2019), 6.
[37] L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.
[38] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell, *Temporal collaborative filtering with Bayesian probabilistic tensor factorization*, in Proceedings of the 10th SIAM International Conference on Data Mining, SIAM, 2010, pp. 211–222,

https://doi.org/10.1137/1.9781611972801.19.

[39] Z. Xu, F. Yan, and Y. Qi, *Bayesian nonparametric models for multiway data analysis*, IEEE Trans. Pattern Anal. Mach. Intell., 37 (2015), pp. 475–487.

[40] Q. Zhao, L. Zhang, and A. Cichocki, *A tensor-variate Gaussian process for classification of multidimensional structured data*, in Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI 2013, pp. 1041–1047.

[41] Q. Zhao, L. Zhang, and A. Cichocki, *Bayesian CP factorization of incomplete tensors with automatic rank determination*, IEEE Trans. Pattern Anal. Mach. Intell., 37 (2015), pp. 1751–1763.

[42] Q. Zhao, L. Zhang, and A. Cichocki, *Bayesian Sparse Tucker Models for Dimension Reduction and Tensor Completion*, preprint, https://arxiv.org/abs/1505.02343, 2015.

[43] Q. Zhao, G. Zhou, L. Zhang, A. Cichocki, and S. I. Amari, *Bayesian robust tensor factorization for incomplete multiway data*, IEEE Trans. Neural Netw. Learn. Syst., 27 (2016), pp. 736–748.