KU LEUVEN

STADIUS
Center for Dynamical Systems, Signal Processing and Data Analytics

Citation/Reference	Vervliet, N., Debals, O., Sorber, L., De Lathauwer, L. (2014), **Breaking the Curse of Dimensionality Using Decompositions of Incomplete Tensors: Tensor-based scientific computing in big data analysis** Signal Processing Magazine, IEEE, 31 (5), 71-79.
Archived version	Author manuscript: the content is identical to the content of the published paper, but without the final typesetting by the publisher
Published version	http://dx.doi.org/10.1109/MSP.2014.2329429
Journal homepage	http://www.signalprocessingsociety.org/publications/periodicals/spm/
Author contact	Nico.Vervliet@esat.kuleuven.be *+ 32 (0)16 3 20362*
IR	https://lirias.kuleuven.be/handle/123456789/455978

(article begins on next page)

Breaking the Curse of Dimensionality using Decompositions of Incomplete Tensors

Nico Vervliet Otto Debals Laurent Sorber Lieven De Lathauwer

Higher-order tensors and their decompositions are abundantly present in domains such as signal processing (e.g. higher-order statistics [1], sensor array processing [2]), scientific computing (e.g. discretized multivariate functions [3]–[6]) and quantum information theory (e.g. representation of quantum many-body states [7]). In many applications the, possibly huge, tensors can be approximated well by compact multilinear models or *decompositions*. Tensor decompositions are more versatile tools than the linear models resulting from traditional matrix approaches. Compared to matrices, tensors have at least one extra dimension. The number of elements in a tensor increases exponentially with the number of dimensions, and so do the computational and memory requirements. The exponential dependency (and the problems that are caused by it) is called the *curse of dimensionality*. The curse limits the order of the tensors that can be handled. Even for modest order, tensor problems are often large-scale. Large tensors can be handled, and the curse can be alleviated or even removed, by using a decomposition that represents the tensor, instead of the tensor itself. However, most decomposition algorithms require full tensors, which renders these algorithms infeasible for large datasets. If a tensor can be represented by a decomposition, this hypothesized structure can be exploited by using *compressed sensing* type methods working on *incomplete* tensors, i.e. tensors with only a few known elements.

In domains such as scientific computing and quantum information theory, tensor decompositions such as the Tucker decomposition and tensor trains have successfully been applied to represent large tensors. In the latter case, the tensor can contain more elements than the number of atoms in the universe [8] (estimated at $\mathcal{O}(10^{82})$). Algorithms to compute these decompositions

Contact: Nico.Vervliet@esat.kuleuven.be

Full reference: Vervliet, N., Debals, O., Sorber, L., De Lathauwer, L., "Breaking the Curse of Dimensionality Using Decompositions of Incomplete Tensors: Tensor-based scientific computing in big data analysis," Signal Processing Magazine, IEEE, vol.31, no.5, pp.71–79, Sept. 2014 (doi: 10.1109/MSP.2014.2329429)

©2014 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

using only a few *mode-n vectors* of the tensors have been developed to cope with the curse of dimensionality. In this tutorial, we show on the one hand how decompositions already known in signal processing (e.g. the canonical polyadic decomposition and the Tucker decomposition) can be used for large and incomplete tensors, and on the other hand how existing decompositions and techniques from scientific computing can be used in a signal processing context. We conclude with a convincing proof-of-concept case study from materials sciences, to the authors' knowledge the first known example of breaking the curse of dimensionality in data analysis.

NOTATION AND PRELIMINARIES

A general N-th order tensor of size $I_1 \times I_2 \times \cdots \times I_N$ is denoted by a calligraphic letter as $\mathcal{A} \in \mathbb{C}^{I_1 \times I_2 \times \cdots \times I_N}$, and is a multidimensional array of numerical values $a_{i_1 i_2 \cdots i_N} = \mathcal{A}(i_1, i_2, \ldots, i_N)$. Tensors can be seen as a higher-order generalization of vectors (denoted by a bold, lowercase letter, e.g. $\mathbf{a}$) and matrices (denoted by a bold, uppercase letter, e.g. $\mathbf{A}$). In the same way as matrices have rows and columns, tensors have mode-n vectors which are constructed by fixing all but one index, e.g. $\mathbf{a} = \mathcal{A}(i_1, \ldots, i_{n-1}, :, i_{n+1}, \ldots, i_N)$. The mode-1 vectors are the columns of the tensor, the mode-2 vectors are the rows of the tensor and so on. More generally, an n-th order slice is constructed by fixing all but n indices. Tensors often need to be *reshaped*. An example is the mode-n matrix unfolding of a tensor $\mathcal{A}$ which arranges the mode-n vectors in a certain order as the columns of a matrix $\mathbf{A}_{(n)}$ [9], [10].

A number of products have to be defined when working with tensors. The outer product of two tensors $\mathcal{A} \in \mathbb{C}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{B} \in \mathbb{C}^{J_1 \times J_2 \times \cdots \times J_M}$ is given as $(\mathcal{A} \otimes \mathcal{B})_{i_1 i_2 \cdots i_N j_1 j_2 \cdots j_M} = a_{i_1 i_2 \cdots i_N} b_{j_1 j_2 \cdots j_M}$. The mode-$n$ tensor-matrix product between a tensor $\mathcal{A} \in \mathbb{C}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $\mathbf{B} \in \mathbb{C}^{J \times I_n}$ is defined as $(\mathcal{A} \cdot_n \mathbf{B})_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} a_{i_1 i_2 \cdots i_N} b_{j i_n}$. The Hadamard product $\mathcal{A} * \mathcal{B}$ for $\mathcal{A}, \mathcal{B} \in \mathbb{C}^{I_1 \times I_2 \times \cdots \times I_N}$ is the element-wise product. Finally, the Frobenius norm of a tensor $\mathcal{A}$ is denoted by $||\mathcal{A}||$ [9], [10].

TENSOR DECOMPOSITIONS

Most tensors of practical interest in applications are generated by some sort of process, e.g. a partial differential equation, a signal measured on a multidimensional grid, or the interactions between atoms. The resulting structure can be exploited by using decompositions which approximate the tensor using only a small number of parameters. By using tensor decompositions

instead of full tensors, the curse of dimensionality can be alleviated or even removed. We look into three decompositions in this tutorial: the canonical polyadic decomposition, the Tucker decomposition and the tensor train decomposition. We conclude with a more general concept from scientific computing and quantum information theory called tensor networks. For more theory and applications we direct the reader to the references, especially [4]–[6], [9]–[11].

Canonical polyadic decomposition

In a *polyadic decomposition* (PD), a tensor $\mathcal{T}$ is written as a sum of R rank-1 tensors, each of which can be written as the outer product of N factor vectors $\mathbf{a}_r^{(n)}$:

$$\mathcal{T} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \otimes \mathbf{a}_r^{(2)} \otimes \cdots \otimes \mathbf{a}_r^{(N)} \triangleq \left[\!\!\left[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)} \right]\!\!\right]. \tag{1}$$

The latter notation is a shorthand for the PD and the factor vectors $\mathbf{a}_r^{(n)}$ are the columns of the factor matrices $\mathbf{A}^{(n)}$ [9]. The PD is called *canonical* (CPD) when R is the minimum number of rank-1 terms needed for (1) to be exact. In this case, R is the CP rank of the tensor. Ignoring the trivial indeterminacies due to scaling and ordering of the rank-1 terms, the CPD is unique under mild conditions [12]. The decomposition has many names such as PARAFAC (chemometrics), CANDECOMP (psychometrics) or R-term representation (scientific computing) [4], [9].

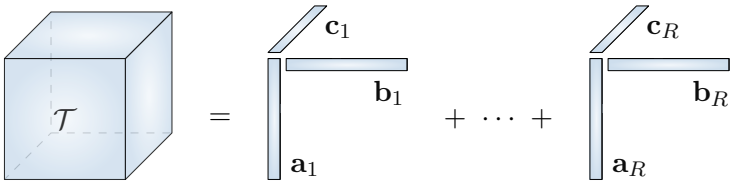


Fig. 1. A *polyadic decomposition* of a third-order tensor $\mathcal{T}$ takes the form of a sum of R rank-1 tensors. If R is the minimum number for the equality to hold, the decomposition is called *canonical*, and R is the rank of the tensor.

The number of free parameters in this decomposition is only $R\left(\left(\sum_{n=1}^{N} I_n\right) - N + 1\right)$ (because of the scaling indeterminacy) which is $\mathcal{O}(NIR)$ assuming $I_n = I$, $n = 1, \ldots, N$. More importantly, it is linear in the number of dimensions N. This means the curse of dimensionality can be broken by using a CPD instead of a full tensor if the tensor admits a good CPD [13]. In many practical cases in signal processing R is low and $R \ll I$. In cases where the rank R cannot be derived from the problem definition, finding the rank is a hard problem. In practice, many CPDs will be fitted to the data until a sufficiently low approximation error is attained

[13]. There is however no guarantee that this process yields the CP rank R, as the best rank-R approximation may not exist. This is due to the fact that the set of rank-R tensors is not closed, which means a sequence of rank-R' tensors with $R' < R$ can converge to a rank-R tensor while two or more terms grow without bounds. This problem is referred to as degeneracy [9], [14], [15]. By imposing constraints such as non-negativity or orthogonality on the factor matrices, degeneracy can be avoided [10], [14], [16].

Tucker decomposition and Low Multilinear Rank Approximation

The Tucker decomposition of a tensor $\mathcal{T}$ is given as a multilinear transformation of a typically small core tensor $\mathcal{G} \in \mathbb{C}^{R_1 \times R_2 \times \cdots \times R_N}$ by factor matrices $\mathbf{A}^{(n)} \in \mathbb{C}^{I_n \times R_n}$, $n = 1, \ldots, N$:

$$\mathcal{T} = \mathcal{G} \cdot_1 \mathbf{A}^{(1)} \cdot_2 \mathbf{A}^{(2)} \cdots \cdot_N \mathbf{A}^{(N)} \triangleq \left[\!\left[\mathcal{G}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)} \right]\!\right], \tag{2}$$

where the latter is a shorthand notation [9]. The N-tuple $(R_1, R_2, \ldots, R_N)$ for which the core size is minimal is called the multilinear rank. R_1 is the dimension of the column space, R_2 is the dimension of the row space, and more generally, R_n is the dimension of the space spanned by the mode-n vectors [9]. In general, the Tucker decomposition (2) is not unique, but the subspaces spanned by the vectors in the factor matrices are, which is useful in certain applications [9], [10]. In its original definition, the Tucker decomposition imposed orthogonality and ordering constraints on the factor matrices and the core tensor. In this definition, the Tucker decomposition can be interpreted as an higher-order generalization of the singular value decomposition (SVD) and can be obtained by reliable algorithms from numerical linear algebra (in particular algorithms for computing the SVD). In this context the names *multilinear SVD* (MLSVD) and *higher-order SVD* (HOSVD) are also used [17].

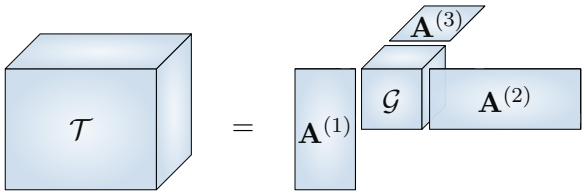


Fig. 2. The Tucker decomposition of a third-order tensor $\mathcal{T}$ involves a multilinear transformation of a core tensor $\mathcal{G}$ by factor matrices $\mathbf{A}^{(n)}$, $n = 1, \ldots, N$.

The number of parameters in the Tucker decomposition is $\mathcal{O}\left(NIR + R^N\right)$ when we take $I_n =$

I and $R_n = R$, $n = 1, \ldots, N$. This means the number of parameters in a Tucker decomposition still depends exponentially on the number of dimensions N. The curse of dimensionality is alleviated, however, as typically $R \ll I$. More generally, when a tensor is *approximated* by (2) where the size of the core tensor is chosen by the user, this decomposition is called a *low multilinear rank approximation* (LMLRA). As in PCA, a Tucker decomposition can be compressed or truncated by omitting small multilinear singular values [9], [10]. This reduction in R is beneficial given the exponential factor $\mathcal{O}\left(R^N\right)$ in the number of parameters, as the total number of parameters decreases exponentially. Note that truncated Tucker decomposition is just one, not necessarily optimal, way to obtain an LMLRA [17].

Tensor trains

Tensor trains (TT) are a concept from scientific computing and from quantum information theory where it is known as *matrix product states* (MPS) [3], [5]–[7]. Each element in a tensor $\mathcal{T}$ can be written as

$$t_{i_1 i_2 \cdots i_N} = \sum_{r_1, r_2, \ldots, r_{N-1}} a^{(1)}_{i_1 r_1} a^{(2)}_{r_1 i_2 r_2} \cdots a^{(N)}_{r_{N-1} i_N},$$

with $r_n = 1, \ldots, R_n$, $n = 1, \ldots, N-1$. The matrices $\mathbf{A}^{(1)} \in \mathbb{C}^{I_1 \times R_1}$ and $\mathbf{A}^{(N)} \in \mathbb{C}^{R_{N-1} \times I_N}$ are the 'head' and 'tail' of the train; the core tensors $\mathcal{A}^{(n)} \in \mathbb{C}^{R_{n-1} \times I_n \times R_n}$, $n = 2, \ldots, N-1$, are the 'carriages' as can be seen in Figure 3. The auxiliary indices R_n, $n = 1, \ldots, N-1$ are called the *compression ranks* or the TT ranks [3]. It can be proven that the compression ranks are bounded by the CP rank of a tensor [18].

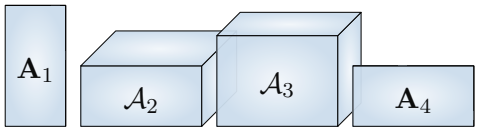


Fig. 3. A fourth-order tensor $\mathcal{T}$ can be written as a *tensor train* by linking a matrix $\mathbf{A}^{(1)}$, two tensors $\mathcal{A}^{(2)}$ and $\mathcal{A}^{(3)}$ (the carriages) and a matrix $\mathbf{A}^{(4)}$.

A TT combines the good properties of the CPD and the Tucker decomposition. The number of parameters in a TT is $\mathcal{O}\left(2IR + (N-2)IR^2\right)$ assuming $I_n = I$, $R_n = R$, $n = 1, \ldots, N$, which is linear in the number of dimensions, similar to a CPD [3], [6]. This means a TT is suitable for high-dimensional problems, as using it removes the curse of dimensionality. As for the Tucker

decomposition, numerically reliable algorithms such as the SVD can be used to compute the decomposition [3], [17].

Tensor networks

The TT decomposition represents a higher-order tensor as a set of linked (lower-order) tensors and matrices, and is an example of a linear *tensor network*. A more general tensor network is a set of interconnected tensors. This can be visualized using *tensor network diagrams* (see Figure 4) [4], [7]. Each vector, matrix or tensor is represented as a dot. The order of each tensor is determined by the number of edges connected to it. An interconnection between two dots represents a *contraction*, which is the summation of the products over a common index. Tensor network diagrams are an intuitive and visual way to efficiently represent decompositions of higher-order tensors. An example is the *hierarchical Tucker* (HT) decomposition (see Figure 4), which is another important decomposition used in scientific computing [4]–[6]. More complicated tensor networks can also contain cycles, e.g. *tensor chains* and *projected entangled-pair states* (PEPS) from quantum physics [5], [7].

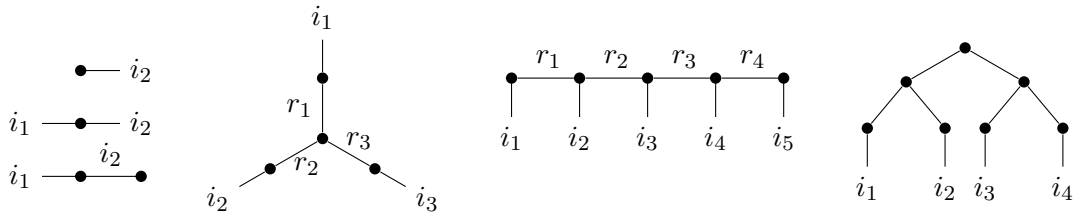


Fig. 4. Different types of tensor networks. From left to right: a vector, a matrix and their matrix-vector product (a *contraction*); a Tucker decomposition; a TT decomposition; and a HT decomposition.

COMPUTING DECOMPOSITIONS OF LARGE, INCOMPLETE TENSORS

To compute tensor decompositions, most algorithms require a full tensor and are therefore not an option for large and high-dimensional datasets. The knowledge that the data is structured and can be represented by a small number of parameters can be exploited by sampling the tensor in only a few elements. Then, the decomposition is calculated using an *incomplete* tensor. There are two important situations where incomplete datasets are used. In the first case some elements are unknown, e.g. because of a broken sensor [19], or unreliable, e.g. because of Rayleigh scattering [15], and the matrix or tensor needs to be *completed* [20]. In the second case, the cost of acquiring

a full tensor is too high in terms of money, time or storage requirements. By sampling the tensor in only a few elements, this cost can be reduced.

Compressed sensing (CS) methods are used to reconstruct signals using only a few measurements taken by a linear projection of the original dataset [21]. Many extensions of these methods to tensors have been developed [10], [22] and new methods tailored to tensors have emerged, e.g. [23], [24]. In this tutorial we focus on a class of CS methods where decompositions of very large tensors are computed using only a small number of known elements. In particular, we first discuss methods to compute a CPD from a randomly sampled incomplete tensor. Then we discuss how matrices can be approximated by extracting only a few rows and columns. This idea can be extended to tensors, and we conclude by elaborating on two mode-n vector sampling methods: one for the TT decomposition and one for the LMLRA.

Optimization-based algorithms

Most algorithms to compute a CPD use optimization to find the factor matrices $\mathbf{A}^{(n)}$:

$$\min_{\mathbf{A}^{(1)},\mathbf{A}^{(2)},\ldots,\mathbf{A}^{(N)}} \frac{1}{2} \left|\left| \mathcal{T} - \left[\!\left[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)} \right]\!\right] \right|\right|^2, \tag{3}$$

which is a least squares problem in each factor matrix separately. The popular *alternating least squares* (ALS) method alternately solves a least squares problem for one factor matrix while fixing the others. This method is easy to implement and works well in many cases, but has a linear convergence rate and tends to be slow when the factor vectors become more aligned. It is even possible that the algorithm does not converge at all [25], [26]. CP-OPT uses a non-linear conjugate gradients method to solve (3) [26]. By using first-order information, the method also achieves linear convergence. Recently, some new methods based on non-linear least squares (NLS) algorithms have been developed. These methods exploit the structure in the objective function's approximate Hessian. Due to the NLS framework, second order convergence can be attained under certain circumstances [25], [27]. The latter two methods are both guaranteed to converge to a stationary point, which can be a local optimum, however.

Although efficient methods exist, the complexity of all methods working on full tensors is at least $\mathcal{O}\left(I^N\right)$, which becomes infeasible for large, high-dimensional tensors. To handle missing

data, problem (3) can be adapted to

$$\min_{\mathbf{A}^{(1)},\mathbf{A}^{(2)},\ldots,\mathbf{A}^{(N)}} \frac{1}{2} \left|\left| \mathcal{W} * \left(\mathcal{T} - \left[\!\left[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)} \right]\!\right] \right) \right|\right|^2,$$

where $\mathcal{W} \in \{0,1\}^{I_1 \times I_2 \times \cdots \times I_N}$ is a binary observation tensor with a 1 for every known element [19]. The popular ALS method has been extended by using an *expectation maximization* (EM) framework to impute each missing value with a value from the current CP model [15]. Because of the imputation, the ALS-EM method still suffers from curse of dimensionality. The CP-WOPT method is an extension of the CP-OPT method and uses only the known elements, thereby relaxing the curse of dimensionality [19]. Adaptions of the Jacobian and the Gramian of the Jacobian for incomplete tensors can be used in an inexact non-linear least squares framework. Second order convergence can again be attained under certain circumstances, while the computational complexity is still linear in the number of known elements [28].

The distribution of the known elements in the tensor can be random, although performance may decrease in case of missing mode-n vectors or slices [19], [28]. The elements should be known a priori, contrary to the mode-n vector based algorithms. Constraints such as non-negativity or a Vandermonde structure can easily be added to the factor matrices, which is useful for many signal processing applications [16].

Pseudo-skeleton approximation for matrices

Instead of randomly sampling a tensor, a more drastic approach can be taken by sampling only mode-n vectors and only using these mode-n vectors in the decomposition. These techniques originate from the *pseudo-skeleton approximation* or the *CUR* decomposition for matrices. These decompositions state that a matrix $\mathbf{A} \in \mathbb{C}^{I_1 \times I_2}$ of rank R can be approximated using only R columns and R rows of this matrix:

$$\mathbf{A} = \mathbf{CGR}, \tag{4}$$

where $\mathbf{C} \in \mathbb{C}^{I_1 \times R}$ has R columns with indices $\mathcal{J}$ of $\mathbf{A}$, i.e. $\mathbf{C} = \mathbf{A}(:, \mathcal{J})$, $\mathbf{R} \in \mathbb{C}^{R \times I_2}$ has R rows with indices $\mathcal{I}$ of $\mathbf{A}$, i.e. $\mathbf{R} = \mathbf{A}(\mathcal{I}, :)$ and $\mathbf{G} = \hat{\mathbf{A}}^{-1}$, where $\hat{\mathbf{A}}$ contains the intersection of $\mathbf{C}$ and $\mathbf{R}$, i.e. $\hat{\mathbf{A}} = \mathbf{A}(\mathcal{I}, \mathcal{J})$. If $\text{rank}(\mathbf{A}) = R$, then (4) is exact when $\mathbf{C}$ has R linearly independent columns of $\mathbf{A}$ and $\mathbf{R}$ has R linearly independent rows of $\mathbf{A}$ (which implies that $\hat{\mathbf{A}}$

is non-singular) [8]. Usually we are interested in the case where rank$(\mathbf{A}) > R$. The best choice for the submatrix $\hat{\mathbf{A}}$ (and consequently $\mathbf{C}$ and $\mathbf{R}$) is in this case the $R \times R$ submatrix having the largest volume, which is given by the modulus of its determinant [29].

To determine the optimal submatrix $\hat{\mathbf{A}}$, the determinants of all possible submatrices have to be evaluated. This is computationally challenging and, moreover, all the elements of the matrix have to be known. A heuristic called *cross approximation* (CA) can be used to calculate a quasi-optimal maximal volume submatrix by only looking at a few rows and columns. The following general scheme can be used (based on [30]). An initial column index set $\mathcal{J} \subset \{1, 2, \ldots, I_2\}$ and an initial row index set $\mathcal{I} \subset \{1, 2, \ldots, I_1\}$ are chosen and $\mathbf{C}$ is defined as $A(:, \mathcal{J})$. Then the submatrix $\hat{\mathbf{C}} = \mathbf{C}(\hat{\mathcal{I}}, :)$ with (approximately) the largest volume is calculated, for example using a technique based on full pivoting [30]. Next, the process is repeated for the rows, i.e. the subset $\hat{\mathcal{J}}$ resulting in the maximal volume submatrix $\hat{\mathbf{R}} = \mathbf{R}(:, \hat{\mathcal{J}})$ in $\mathbf{R} = \mathbf{A}(\mathcal{I}, :)$ is calculated. Next, the index sets are updated as $\mathcal{J} = \mathcal{J} \bigcup \hat{\mathcal{J}}$ and $\mathcal{I} = \mathcal{I} \bigcup \hat{\mathcal{I}}$ and the process is repeated until a stopping criterion is met, e.g. when the norm of the residual $||\mathbf{A} - \mathbf{CGR}||$ is small enough. To calculate the norm, only the extracted rows and columns are taken into account. To make this more concrete, we give a simple method selecting one column and row at a time [31]:

1) Set $\mathcal{J} = \emptyset$, $\mathcal{I} = \emptyset$, $j_1 = 1$ and $p = 1$.
2) Extract the column $\mathbf{A}(:, j_p)$ and find the maximal volume submatrix in the residual, i.e. the largest element in modulus in the vector $\mathbf{a}_{j_p}$ minus the corresponding elements in the already known rank-1 terms, and set i_p to its location.
3) Extract the row $\mathbf{A}(i_p, :)$ and find the maximal volume submatrix in the residual that is not in the previously chosen column j_p, and set j_{p+1} to its location.
4) Set $\mathcal{J} = \mathcal{J} \bigcup \{j_p\}$, $\mathcal{I} = \mathcal{I} \bigcup \{i_p\}$.
5) If the stopping criterion is not satisfied, set $p = p + 1$ and go to step 2.

For more details, we refer the interested reader to [8], [30], [31].

Cross approximation for TT

Before we outline a CA based algorithm for the TT decomposition, we first present a simplified version of a TT algorithm for full tensors based on repeated truncated SVD [8]:

1) Set $R_0 = 1$ and $\mathbf{M} = \text{reshape}\left(\mathcal{T}, \left[I_1, \prod_{k=2}^{N} I_k\right]\right)$.

2) For $n = 1, \ldots, N-1$:

   a) Calculate the (truncated) SVD: $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{H}}$ (with $^{\mathrm{H}}$ the conjugated transpose).

   b) Set $\mathcal{A}^{(n)} = \text{reshape}\left(\mathbf{U}, [R_{n-1}, I_n, R_n]\right)$ and if $n < N-1$, set $\mathbf{M} = \text{reshape}(\mathbf{\Sigma}\mathbf{V}^{\mathrm{H}}, [R_n I_{n+1}, \prod_{k=n+2}^{N} I_k])$.

3) $\mathbf{A}^{(N)} = \mathbf{\Sigma}\mathbf{V}^{\mathrm{H}}$.

The truncation step 2a determines the compression rank R_n. In a scientific computing context, the compression ranks R_n are chosen such that the decomposition approximates the (noise free) tensor with a user-defined accuracy ϵ [3]. In signal processing, the tensor is often perturbed by noise. Therefore, the compression ranks R_n can be determined by using a procedure estimating the noise level.

The use of the SVD in 2a has two disadvantages: all the elements in the tensor need to be known and when this tensor is large, calculating the SVD is expensive. In [8] a CA type method is suggested: the SVD can be replaced by a pseudo-skeleton approximation as described above by replacing $\mathbf{U}$ with $\mathbf{CG}$ and $\mathbf{\Sigma}\mathbf{V}^{\mathrm{H}}$ with $\mathbf{R}$. The matrix $\mathbf{R}$ does not require extra calculations and working with $\mathbf{R}$ does not require additional memory as it can be handled implicitly by selecting the proper indices. This algorithm requires the compression ranks to be known in advance. By using a compression or *rounding* algorithm on the resulting TT decomposition, the compression rank can be overestimated safely (see e.g. [3] for a compression method based on the truncated SVD). In a signal processing context, this rounding algorithm can be adapted to use a noise level estimation procedure instead of using a user-defined accuracy ϵ. For practical implementation details we refer to [8].

The positions of the elements needed by the CA algorithm are unknown a priori, but are generated based on information in the mode-n vectors that already have been extracted. In a scientific computing context, where the tensor often is given as a multivariate function, this is not a problem as sampling an entry is evaluating this function. In a signal processing context, this means that either the elements are sampled while running the CA algorithm, or the full tensor has to be known a priori. This last condition can be relaxed, however, by *imputing* unknown elements in selected mode-n vector by an estimate of the value of these elements, e.g. the mean value over the mode-n vector. This, however, only works well if the imputed value effectively is a good estimator of the unknown value. Only $\mathcal{O}\left(2KNR\right)$ columns of length $R_{n-1}I_n$ are investigated

during the CA algorithm, where K is the number of iterations in the CA algorithm and assuming $R_n = R$, $n = 1, \ldots, N-1$. If the compression ranks and the number of iterations are low, very few elements need to be sampled.

Cross approximation for LMLRA

The CA method for TT essentially only replaced the SVD with a pseudo-skeleton approximation. In case of the LMLRA we look at another generalization of the pseudo-skeleton approximation method: $\mathcal{T}$ can be approximated by $\hat{\mathcal{T}} = \left[\!\left[\mathcal{G}; \mathbf{C}^{(1)}\mathbf{G}^{\dagger}_{(1)}, \ldots, \mathbf{C}^{(N)}\mathbf{G}^{\dagger}_{(N)}\right]\!\right]$ where $\mathbf{C}^{(n)}$ contains $\prod_{m\neq n} R_m$ mode-n vectors for $n = 1, \ldots, N$ and where the size of the core tensor $\mathcal{G}$ is $R_1 \times \cdots \times R_N$. The core tensor is the subtensor of $\mathcal{T}$ containing the intersection of the selected mode-n vectors. More concretely, we define the index sets $\mathcal{I}^{(n)} \subset \{1, \ldots, I_n\}$, $n = 1, \ldots, N$. Each column of $\mathbf{C}^{(n)}$ contains a selected mode-n vector defined by an index set in $\bigtimes_{m\neq n}\mathcal{I}^{(m)}$, i.e. $\mathcal{T}(i_1, \ldots, i_{n-1}, :, i_{n+1}, \ldots, i_N)$ with $(i_1, \ldots, i_{n-1}, i_{n+1}, \ldots, i_N) \in \bigtimes_{m\neq n}\mathcal{I}^{(m)}$. $\mathbf{C}^{(n)}$ thus is a matricized $(R_1 \times \cdots \times R_{n-1} \times I_n \times R_{n+1} \times \cdots \times R_N)$ subtensor of $\mathcal{T}$. The intersection core tensor then is defined as $\mathcal{G} = \mathcal{T}(\mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(N)})$. To determine the index sets $\mathcal{I}^{(n)}$ an adaptive procedure can be used. Each iteration the index $i^{(n)}$ having the largest modulus of the residual in the mode-n vector through the pivot, is added to $\mathcal{I}^{(n)}$. The residual is defined as $\mathcal{E} = \mathcal{T} - \hat{\mathcal{T}}$ where the matrices $\mathbf{C}^{(n)}$, $n = 1, \ldots, N$ and the core tensor $\mathcal{G}$ are defined by the current index sets $\mathcal{I}^{(n)}$. A simplified version of this *fiber sampling tensor decomposition algorithm* [32] is given below:

1) Choose an initial mode-N vector in $\mathcal{T}$ defined by $i_1^{(n)}$, for $n = 1, \ldots, N-1$ and set $i_1^{(N)}$ to the index containing the maximal modulus in this fiber.
2) Set $\mathcal{I}^{(n)} = \{i_1^{(n)}\}$ for $n = 1, \ldots, N$ and set the pivot to $(i_1^{(1)}, \ldots, i_1^{(N)})$.
3) For $r = 2, \ldots, R$:
    a) For each mode $n = 1, \ldots, N$:
        i) Select the index $i_r^{(n)}$ of the maximal modulus of the mode-n vector $\mathbf{e}$ going through the pivot in the residual tensor[1] $\mathcal{E}$, i.e. in $\mathbf{e} = \mathcal{E}(i_r^{(1)}, \ldots, i_r^{(n-1)}, :, i_{r-1}^{(n+1)}, \ldots, i_{r-1}^{(N)})$.
        ii) Set $\mathcal{I}^{(n)} = \mathcal{I}^{(n)} \bigcup \{i_r^{(n)}\}$ and select $(i_r^{(1)}, \ldots, i_r^{(n)}, i_{r-1}^{(n+1)}, \ldots, i_{r-1}^{(N)})$ as new pivot.

[1]Unless for $r = 2$ and $n = 1$, then we select the maximal modulus in $\mathcal{T}$ [32].

Each matrix $\mathbf{C}^{(n)}$ contains $|\times_{m\neq n}\mathcal{I}^{(m)}| = \mathcal{O}\left(R^{N-1}\right)$ columns. The total number of mode-n vectors of length I_n that has to be extracted in this algorithm is then $\mathcal{O}\left(NR^{N-1}\right)$. Similarly to the pseudo-skeleton approximation, an exact decomposition based on fiber sampling can be attained if $\mathcal{T}$ has an exact LMLRA structure and multilinear rank $(R_1,\ldots,R_N)$, i.e. $\mathcal{T} = [\![\mathcal{G};\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)}]\!]$. In this case, it can be proven that only R_n mode-n fibers per mode n and $\prod_{n=1}^{N} R_n$ core elements have to be extracted [32]. In both cases, the computational complexity still has an exponential dependence on the number of dimensions [32]. Even with CA, the representation of a tensor as an LMLRA is limited by the curse of dimensionality to low-dimensional problems. We can make the same remarks for this method as for the TT decomposition concerning the fact that the indices of the required elements are only known at runtime. A variation of this algorithm determines the largest element in slices instead of in mode-n vectors [31]. An alternative to the pseudo-skeleton approach, is to sample mode-n vectors after a fast estimation of probability densities [33].

CASE STUDIES

To illustrate the use of the decompositions and incomplete tensors, two case studies are reported. The first case study shows how the concepts can be applied in a signal processing context. To compare the results with full tensor methods, moderate size tensors are used. The second case study gives an example from materials sciences, where a huge tensor is decomposed while using only a very small fraction of the elements.

Multidimensional harmonic retrieval

Multidimensional harmonic retrieval (MHR) problems appear frequently in signal processing, e.g. in radar applications and channel sounding [34]. To model a multipath wireless channel, for example, a broadband wireless channel sounder can be used to measure a (time-varying) channel in the time, frequency and spatial domains. The measurement data can then be transformed into a tensor:

$$y_{i_1 i_2 \cdots i_D k} = \sum_{r=1}^{R} s_r(k) \prod_{d=1}^{D} \mathrm{e}^{\mathrm{j}(i_d-1)\mu_r^{(d)}} + n_{i_1 i_2 \cdots i_D k}, \tag{5}$$

where $\mathrm{j}^2 = -1$ and $s_r(k)$ is the k-th complex symbol carried by the r-th multidimensional harmonic. The parameters $\mu_r^{(d)}$ are for example the direction of departure, the direction of arrival,

the Doppler shift, the delay, and so on. (For more information we refer the interested reader to [34].) The noise $n_{i_1 i_2 \cdots i_D k}$ is modeled as zero mean i.i.d. additive Gaussian noise. We can rewrite the model as a CPD

$$\mathcal{Y} = \left[\!\left[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(D)}, \mathbf{S} \right]\!\right] + \mathcal{N} \tag{6}$$

with the Vandermonde structured factor matrices $\mathbf{A}^{(d)} \in \mathbb{C}^{I_d \times R}$, $a_{i_d,r}^{(d)} = \mathrm{e}^{\mathrm{j}(i_d-1)\mu_r^{(d)}}$ and $\mathbf{S} \in \mathbb{C}^{K \times R}$ with $s_{kr} = s_r(k)$. In the noiseless case, the CP rank of this tensor is equal to R. The multilinear ranks and the TT ranks are at most R. Uniqueness properties of (6) are given in [35].

To estimate the parameters $\mu_r^{(d)}$, a subspace based approach is used. First, $\mathcal{Y}$ is decomposed using a CPD, an LMLRA or a TT. Then, in case of the LMLRA and TT, we compute the subspaces $\mathbf{B}^{(d)}$ spanned by the mode-d vectors, $d = 1, \ldots, D$. (The parameters can be estimated directly from the factor vectors in case of CPD.) Finally, we use a standard *total least squares* method to estimate the parameters $\hat{\boldsymbol{\mu}}^{(d)}$ from these subspaces (see [36]). Here, we focus on the first two steps, i.e. the approximation of the full or incomplete tensor $\mathcal{Y}$ by a CPD, an LMLRA or a TT and the computation of the subspaces.

In case of a CPD, we use the `cpd_nls` method from Tensorlab [25], [37] to get an estimate of the factor matrices $\hat{\mathbf{A}}^{(d)}$, $d = 1, \ldots, D+1$. This method works on both full and incomplete tensors. Here, we can estimate the parameters directly as the generators of the noisy Vandermonde vectors $\mathbf{a}_r^{(d)}$, $d = 1, \ldots, D$, so the computation of the subspaces is not necessary. The number of sampled entries N_{samples} can be chosen by the user (see Table I).

In case of an LMLRA $\left[\!\left[\hat{\mathcal{G}}; \hat{\mathbf{A}}^{(1)}, \ldots, \hat{\mathbf{A}}^{(D+1)} \right]\!\right]$, we first compute the decomposition using `lmlra` from Tensorlab [37], which uses an NLS-based optimization method on the full tensor, and using `lmlra_aca`, which implements a fiber sampling adaptive cross approximation technique. In the latter case, the choice of the core size $R_1 \times \cdots \times R_{D+1}$ controls the number of touched elements, i.e. the number of elements from the tensor that are used during the algorithm (see Table I). Recall that R is the number of multidimensional harmonics in (5). The subspaces $\hat{\mathbf{B}}^{(d)}$ are now computed using the first R left singular vectors of the unfolded product $(\hat{\mathcal{G}} \cdot_d \hat{\mathbf{A}}^{(d)})_{(d)}$. (It can be verified that these are the dominant mode-d vectors of $\left[\!\left[\hat{\mathcal{G}}; \hat{\mathbf{A}}^{(1)}, \ldots, \hat{\mathbf{A}}^{(D+1)} \right]\!\right]$ if the factor matrices are normalized to have orthonormal columns.)

Finally, in case of TT, we compute the TT cores $\hat{\mathbf{A}}^{(1)}$, $\hat{\mathcal{A}}^{(d)}$, $\hat{\mathbf{A}}^{(D+1)}$ using `tt_full`, which

uses the truncated SVD of the full tensor (cf. supra), and using `dmrg_cross`, which uses cross approximation and touches only a limited number of mode-n vectors (cf. supra). Both methods are available in the TT-toolbox (http://spring.inm.ras.ru/osel/). The number of touched elements is controlled by the compression ranks R_n and the number of iterations K (see Table I). The estimates for the subspaces $\hat{\mathbf{B}}^{(1)}$ and $\hat{\mathbf{B}}^{(d)}$ can be computed using the first R left singular vectors of $\hat{\mathbf{A}}^{(1)}$ and of the mode-2 unfolding $\hat{\mathcal{A}}^{(d)}_{(2)}$, $d = 2, \ldots, D$, respectively.

TABLE I
NUMBER OF PARAMETERS AND TOUCHED ELEMENTS FOR THE THREE DECOMPOSITIONS OF INCOMPLETE TENSORS. THE NUMBER OF TOUCHED ELEMENTS CONCERN THE PRESENTED ALGORITHMIC VARIANTS. IN CASE OF A CPD AND TT, THE CURSE OF DIMENSIONALITY CAN BE OVERCOME.

	# Parameters	# Touched elements
CPD	$\mathcal{O}(NIR)$	N_{samples}
LMLRA	$\mathcal{O}\left(NIR + R^N\right)$	$\mathcal{O}\left(NIR^{N-1}\right)$
TT	$\mathcal{O}\left(2IR + (N-2)IR^2\right)$	$\mathcal{O}\left(2KNR^2I\right)$

With two experiments, we show how the number of touched elements and noise influence the quality of the retrieved parameters using the three decompositions. We create an $8 \times 8 \times 8 \times 8 \times 20$ tensor $\mathcal{Y}$ with rank $R = 4$ according to (6). The $D = 4$ parameter vectors are chosen as follows: $\boldsymbol{\mu}^{(1)} = [1.0, -0.5, 0.1, -0.8]$, $\boldsymbol{\mu}^{(2)} = [-0.5, 1.0, -0.9, 1.0]$, $\boldsymbol{\mu}^{(3)} = [0.2, -0.6, 1.0, 0.4]$, $\boldsymbol{\mu}^{(4)} = [-0.8, 0.4, 0.3, -0.1]$. Each of the $R = 4$ uncorrelated binary phase shift keying (BPSK) sources takes $K = 20$ values. To evaluate the quality of the estimates, the *root mean square error* (RMSE) is used:

$$E_{\text{RMS}} = \sqrt{\frac{1}{RD} \sum_{r=1}^{R} \sum_{d=1}^{D} \left(\mu_r^{(d)} - \hat{\mu}_r^{(d)}\right)^2}.$$

For each experiment, the median value over 100 Monte Carlo runs is reported.

In the first experiment, the signal-to-noise ratio (SNR) is fixed to 20 dB while the fraction of missing entries varies (see Figure 5, left). When there are no missing entries, we use the corresponding algorithms for full tensors. All methods then attain a similar accuracy. When the fraction of missing entries is increased, the error E_{RMS} also increases, except for TT where the error remains almost constant, but is higher than for CPD and LMLRA. An increase in the error is expected, as there are fewer noisy samples to estimate the parameters from. For 99% missing entries, the CPD algorithm no longer finds a solution as the number of known entries (820) is

close to the number of free parameters (204). The CPD based method has the best performance. ($\mathcal{Y}$ has a CPD structure to start from).

In the second experiment, the number of known elements is kept between 8% and 12% (remember that it is difficult to control the accesses in an adaptive algorithm) and the SNR is varied (see Figure 5, right). In case of the full tensor methods, E_{RMS} is almost equal for all decompositions, unless for low SNR. In case of the incomplete methods, the CPD based method performs better, especially in the low SNR cases.

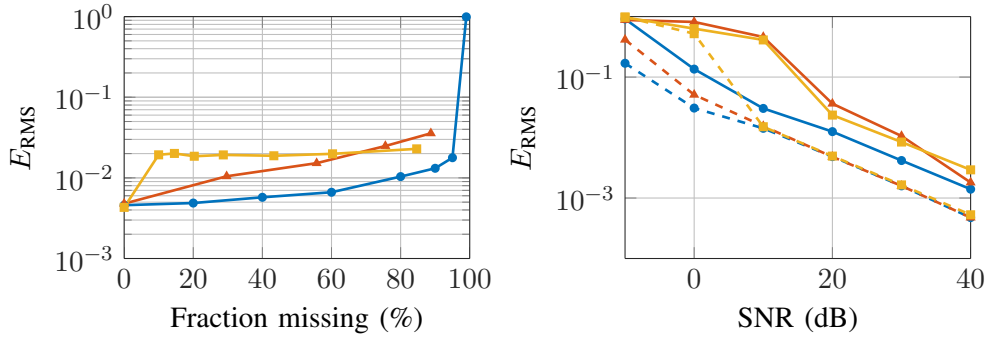


Fig. 5. Influence of the number of known elements (left) and SNR (right) on E_{RMS} for the CPD (—●—), the LMLRA (—▲—) and TT (—■—). The dashed lines give the results for the full tensor methods.

Material sciences example

When designing new materials, the physical properties of these new materials are key parameters. In the case of alloys, the concentrations of the different constituent materials can be used to model the physical properties. In this particular example, we model the melting point of an alloy, using a dataset kindly provided by InsPyro NV, Belgium. The dataset contains a small set of random measurements of the melting point in function of the concentrations of ten different constituent materials. This dataset can be represented as a ninth-order tensor $\mathcal{T}$. (One concentration is superfluous as concentrations must sum to 100%.) The curse of dimensionality is an important problem for this kind of data, as the number of elements in this tensor is approximately $100^N = 10^{18}$ with $N+1$ the number of constituent materials. Because measuring and computing all these elements is infeasible, only 130 000 elements are sampled.

This case study illustrates how a tensor decomposition algorithm for incomplete tensors can overcome the curse of dimensionality. We use the `cpdi_nls` algorithm [28] because it is suitable

for a dataset containing only a small fraction of randomly sampled elements. In particular, we approximate the training tensor $\mathcal{T}_{\text{tr}}$ which contains 70% of the data, by a CPD $\hat{\mathcal{T}} = [\![\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(9)}]\!]$ and repeat this for several ranks R. To evaluate the quality of a rank-R model the validation error E_{val} of the model is computed using an independent validation tensor $\mathcal{T}_{\text{val}}$ containing the remaining 30% of the data. This error is defined as the weighted relative norm of the error between $\mathcal{T}_{\text{val}}$ and the model:

$$E_{\text{val}} = \frac{||\mathcal{W}_{\text{val}} * (\mathcal{T}_{\text{val}} - [\![\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(9)}]\!])||}{||\mathcal{W} * \mathcal{T}_{\text{val}}||}.$$

The binary observation tensor $\mathcal{W}_{\text{val}}$ has only ones at the positions of known validation elements. We also report the 99% quantile of the relative residuals between known elements in the validation set and the model as E_{quant}. The timing experiments are performed on a relatively recent laptop (Intel core i7, quadcore @2.7 GHz, 8 GB Ram, Matlab 2013b).

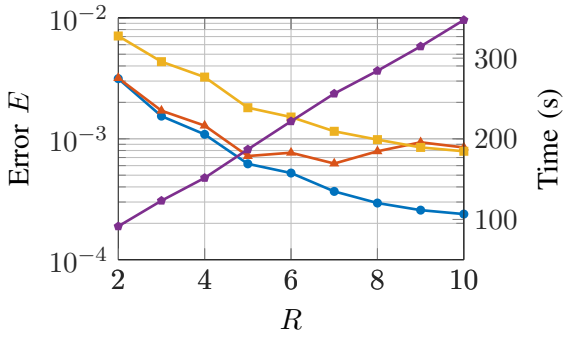


Fig. 6. Errors on training E_{tr} (—●—) and validation E_{val} (—▲—) set and the 99% quantile error E_{quant} (—■—) for different CPDs. The computation time for each model is indicated by (—●—) on the right y-axis.

To compute a CPD from the training tensor, the `cpdi_nls` method [28] is used. This method is an extension of the `cpd_nls` method from Tensorlab [25], [37] for incomplete tensors. When choosing the initial factor matrices, we have to take the high order N into account: from (1) we see that every element in $\hat{\mathcal{T}}$ is the sum of R products of $N = 9$ variables. This means that, if most elements in the factor matrices are close to zero, $\mathcal{T} \approx 0$. Here, we have drawn the elements in the initial factor matrices from a uniform distribution in $(0, 1)$, and we have scaled each factor vector $\mathbf{a}_r^{(n)}$, $n = 1, \ldots, N$ by $\sqrt[N]{\lambda_r}$ where λ_r are the minimizers of $\left|\left|\mathcal{W}_{\text{tr}} * (\mathcal{T}_{\text{tr}} - \sum_{r=1}^{R} \lambda_r \mathbf{a}_r^{(1)} \otimes \cdots \otimes \mathbf{a}_r^{(1)})\right|\right|$. Finally, we use a best out of five strategy, which means that we choose five different optimally scaled initial solutions and keep the best result in terms

of error on the training tensor E_{tr}. The result is shown in Figure 6. Both E_{tr} and E_{quant} keep decreasing as R increases, which indicates that also outliers are modeled when more rank-1 terms are added to the model. Starting from $R = 5$, E_{val} and E_{tr} start diverging, which can indicate that the data is overmodeled for $R > 5$, although E_{quant} keeps decreasing. For the remaining of this case study, we assume $R = 5$ to be a good choice as rank: the relative error E_{quant} is smaller than $1.81 \cdot 10^{-3}$ for 99% of the validation points, while it only took 3 minutes to compute the model. (The time rises linearly in R as can be seen in Figure 6.)

To summarize: by using 10^5 elements we have reduced a dataset containing 10^{18} elements to a model having $NIR \approx 4500$ parameters. We can now go one step further by looking at the values in the different factor vectors (see Figure 7). We see that the factor vectors have a smooth, low degree polynomial-like behavior, a little perturbed by noise. By fitting smooth spline functions to each factor vector, a continuous model for the physical parameter can be created:

$$\mathcal{T} \approx f(c_1, \ldots, c_N) = \sum_{r=1}^{R} \prod_{n=1}^{N} a_r^{(n)}(c_n),$$

where $a_r^{(n)}$ are continuous functions in the concentrations c_n, $n = 1, \ldots, N$. This has many advantages: the high-dimensional model can be visualized more easily and all elements having a certain melting point can be calculated (see for example Figure 8). Furthermore, the model can be used in further steps in the design of the material.

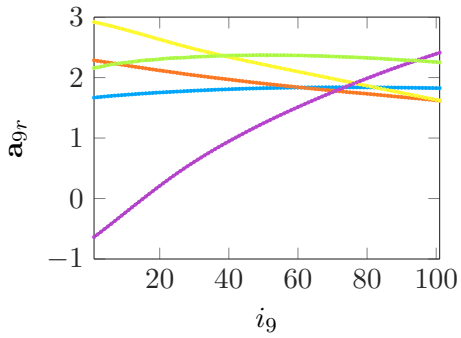


Fig. 7. The values in the $R = 5$ factor vectors follow a smooth function. Here the factor vectors for the ninth mode $\mathbf{a}_r^{(9)}$ are shown as dots.

CONCLUSION

Tensor decompositions open up new possibilities in analysis and computation, as they can alleviate or even break the curse of dimensionality that occurs when working with high-dimensional

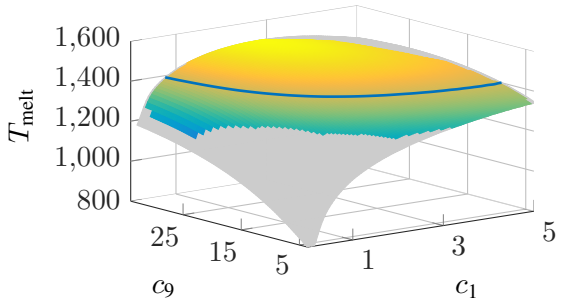


Fig. 8. Visualization of the continuous surface of melting points when all but two concentrations are fixed. The blue line links all points having a melting temperature of 1400 °C. The model is only valid in the colored region.

tensors. Decompositions such as the tensor train decomposition are often used in other fields such as scientific computing and quantum information theory. These decompositions can easily be ported to a signal processing context. We have addressed some problems when computing decompositions of full tensors. By exploiting the structure of a tensor, compressed sensing type methods can be used to compute these decompositions using incomplete tensors. We have illustrated this with random sampling techniques for the CPD, and with mode-n vector sampling techniques originating from scientific computing for the LMLRA and the TT decomposition.

ACKNOWLEDGMENTS

Nico Vervliet is supported by the Research Foundation – Flanders (FWO). The research of Otto Debals and Laurent Sorber is funded by a Ph.D. grant of the Agency for Innovation by Science and Technology (IWT). This research is supported by: (1) Research Council KU Leuven: GOA-MaNet, CoE EF/05/006 Optimization in Engineering (OPTEC), (2) FWO: project G.0427.10N, G.0830.14N, G.0881.14N, (3) the Belgian Federal Science Policy Office: IUAP P7 (DYSCO II, Dynamical systems, control and optimization, 2012-2017), (4) EU: ERC advanced grant no. 339804 (BIOTENSORS). This paper reflects only the authors' views and the Union is not liable for any use that may be made of the contained information.

AUTHORS

Nico Vervliet (Nico.Vervliet@esat.kuleuven.be) obtained the M.Sc. degree in Mathematical Engineering from KU Leuven, Belgium, in 2013. He is a Ph.D. candidate at the STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics of the Electrical Engineering Department (ESAT), KU Leuven and is affiliated with iMinds Medical IT. His research interests

include decomposition algorithms for large and incomplete tensors and for multiview data.

Otto Debals (Otto.Debals@esat.kuleuven.be) obtained the M.Sc. degree in Mathematical Engineering from KU Leuven, Belgium, in 2013. He is a Ph.D. candidate at the STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics of the Electrical Engineering Department (ESAT), KU Leuven and is affiliated with iMinds Medical IT. His research concerns the tensorization of matrix data.

Laurent Sorber (Laurent.Sorber@cs.kuleuven.be) received the M.Sc. and the Ph.D. degrees from the Faculty of Engineering, KU Leuven, Belgium, in 2010 and 2014, respectively. He is affiliated with the STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics of the Electrical Engineering Department (ESAT), with NALAG of the Computer Science Department, KU Leuven, and with iMinds Medical IT. His research includes the development of numerical algorithms for tensor decompositions and structured data fusion. He is the main developer of the Tensorlab toolbox.

Lieven De Lathauwer (Lieven.DeLathauwer@kuleuven-kulak.be) received the Ph.D. degree from the Faculty of Engineering, KU Leuven, Belgium, in 1997. From 2000 to 2007 he was Research Associate with the Centre National de la Recherche Scientifique, France. He is currently Professor with KU Leuven. He is affiliated with the Group Science, Engineering and Technology of Kulak, with the Stadius Center for Dynamical Systems, Signal Processing and Data Analytics of the Electrical Engineering Department (ESAT) and with iMinds Medical IT. He is Associate Editor of the SIAM Journal on Matrix Analysis and Applications and has served as Associate Editor for the IEEE Transactions on Signal Processing. His research concerns the development of tensor tools for engineering applications.

REFERENCES

[1] P. Comon and C. Jutten, *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press, 2010.

[2] N. Sidiropoulos, R. Bro, and G. Giannakis, "Parallel factor analysis in sensor array processing," *IEEE Trans. Signal Process.*, vol. 48, no. 8, pp. 2377–2388, 2000.

[3] I. Oseledets, "Tensor-train decomposition," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2295–2317, 2011.

[4] B. Khoromskij, "Tensors-structured numerical methods in scientific computing: Survey on recent advances," *Chemometrics and Intelligent Laboratory Systems*, vol. 110, no. 1, pp. 1 – 19, 2012.

[5] L. Grasedyck, D. Kressner, and C. Tobler, "A literature survey of low-rank tensor approximation techniques," *ArXiv e-prints*, Feb. 2013.

[6] W. Hackbusch, *Tensor spaces and numerical tensor calculus*, ser. Springer series in computational mathematics. Heidelberg: Springer, 2012, vol. 42.

[7] R. Orus, "A practical introduction to tensor networks: Matrix product states and projected entangled pair states," *ArXiv e-prints*, Jun. 2013.

[8] I. Oseledets and E. Tyrtyshnikov, "TT-cross approximation for multidimensional arrays," *Linear Algebra and its Applications*, vol. 432, no. 1, pp. 70 – 88, 2010.

[9] T. Kolda and B. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.

[10] A. Cichocki, D. Mandic, C. Caiafa, A.-H. Phan, G. Zhou, Q. Zhao, and L. De Lathauwer, "Tensor decompositions for signal processing applications," internal report 14-19, ESAT-STADIUS, KU Leuven (Leuven, Belgium), 2014, Accepted for publication in IEEE Signal Proces. Mag.

[11] P. Comon, "Tensors: A brief introduction," *IEEE Signal Process. Mag.*, vol. 31, no. 3, pp. 44–53, May 2014.

[12] I. Domanov and L. De Lathauwer, "On the uniqueness of the canonical polyadic decomposition of third-order tensors—part II: Uniqueness of the overall decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 3, pp. 876–903, 2013.

[13] G. Beylkin and M. J. Mohlenkamp, "Numerical operator calculus in higher dimensions," *Proceedings of the National Academy of Sciences*, vol. 99, no. 16, pp. 10 246–10 251, 2002.

[14] W. Krijnen, T. Dijkstra, and A. Stegeman, "On the non-existence of optimal solutions and the occurrence of "degeneracy" in the CANDECOMP/PARAFAC model," *Psychometrika*, vol. 73, no. 3, pp. 431–439, 2008.

[15] A. Smilde, R. Bro, P. Geladi, and J. Wiley, *Multi-way analysis with applications in the chemical sciences*. Wiley Chichester, UK, 2004.

[16] L. Sorber, M. Van Barel, and L. De Lathauwer, "Structured data fusion," tech. report 13-177, ESAT-SISTA, KU Leuven (Belgium), 2013.

[17] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[18] I. Oseledets and E. Tyrtyshnikov, "Breaking the curse of dimensionality, or how to use SVD in many dimensions," *SIAM Journal on Scientific Computing*, vol. 31, no. 5, pp. 3744–3759, 2009.

[19] E. Acar, D. Dunlavy, T. Kolda, and M. Mørup, "Scalable tensor factorizations for incomplete data," *Chemometrics and Intelligent Laboratory Systems*, vol. 106, no. 1, pp. 41 – 56, 2011.

[20] S. Gandy, B. Recht, and I. Yamada, "Tensor completion and low-n-rank tensor recovery via convex optimization," *Inverse Problems*, vol. 27, no. 2, p. 025010, 2011.

[21] E. Candès and M. Wakin, "An introduction to compressive sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 21–30, 2008.

[22] C. F. Caiafa and A. Cichocki, "Multidimensional compressed sensing and their applications," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 3, no. 6, pp. 355–380, 2013.

[23] N. Sidiropoulos and A. Kyrillidis, "Multi-way compressed sensing for sparse low-rank tensors," *IEEE Signal Process. Lett.*, vol. 19, no. 11, pp. 757–760, Nov 2012.

[24] Q. Li, D. Schonfeld, and S. Friedland, "Generalized tensor compressive sensing," in *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, July 2013, pp. 1–6.

[25] L. Sorber, M. Van Barel, and L. De Lathauwer, "Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-$(L_r, L_r, 1)$ terms, and a new generalization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 695–720, 2013.

[26] E. Acar, D. Dunlavy, and T. Kolda, "A scalable optimization approach for fitting canonical tensor decompositions," *Journal of Chemometrics*, vol. 25, no. 2, pp. 67–86, 2011.

[27] A.-H. Phan, P. Tichavsky, and A. Cichocki, "Low complexity damped Gauss–Newton algorithms for CANDECOMP/PARAFAC," *SIAM J. Appl. Math.*, vol. 34, no. 1, pp. 126–147, 2013.

[28] O. Debals and N. Vervliet, "Efficiënte tensorgebaseerde methoden voor modellering en signaalscheiding," Master's thesis, KU Leuven, 2013, (Dutch).

[29] S. Goreinov, N. Zamarashkin, and E. Tyrtyshnikov, "Pseudo-skeleton approximations by matrices of maximal volume," *Mathematical Notes*, vol. 62, no. 4, pp. 515–519, 1997.

[30] E. Tyrtyshnikov, "Incomplete cross approximation in the mosaic-skeleton method," *Computing*, vol. 64, pp. 367–380, 2000.

[31] I. Oseledets, D. Savostianov, and E. Tyrtyshnikov, "Tucker dimensionality reduction of three-dimensional arrays in linear time," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 939–956, 2008.

[32] C. Caiafa and A. Cichocki, "Generalizing the column-row matrix decomposition to multi-way arrays," *Linear Algebra and its Applications*, vol. 433, no. 3, pp. 557 – 573, 2010.

[33] M. Mahoney, M. Maggioni, and P. Drineas, "Tensor-CUR decompositions for tensor-based data," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 957–987, 2008.

[34] X. Liu, N. D. Sidiropoulos, and T. Jiang, "Multidimensional harmonic retrieval with applications in MIMO wireless channel sounding," in *Space-Time Processing for MIMO Communications*, A. Gershman and N. Sidiropoulos, Eds. John Wiley & Sons, Ltd, 2005.

[35] M. Sorensen and L. De Lathauwer, "Multidimensional harmonic retrieval via coupled canonical polyadic decomposition," internal report 13-240, ESAT-SISTA, KU Leuven (Leuven, Belgium), 2013.

[36] S. Van Huffel, H. Chen, C. Decanniere, and P. Vanhecke, "Algorithm for time-domain NMR data fitting based on total least squares," *Journal of Magnetic Resonance, Series A*, vol. 110, no. 2, pp. 228 – 237, 1994.

[37] L. Sorber, M. Van Barel, and L. De Lathauwer, "Tensorlab v2.0," January 2014. [Online]. Available: www.tensorlab.net