Linear Algebra and its Applications 435 (2011) 641–658

Contents lists available at ScienceDirect

Linear Algebra and its Applications

journal homepage: www.elsevier.com/locate/laa

Factorization strategies for third-order tensors☆

Misha E. Kilmer [a,*], Carla D. Martin [b]

[a] Mathematics, Tufts University, 113 Bromfield-Pearson Bldg., Medford, MA 02155, United States
[b] Mathematics and Statistics, James Madison University, 112 Roop Hall, MSC 1911, Harrisonburg, VA 22807, United States

ARTICLE INFO

Article history:
Available online 25 October 2010

Submitted by V. Mehrmann

AMS classification:
15A69
65F30

Keywords:
Multilinear algebra
Tensor decomposition
Singular value decomposition
Multidimensional arrays

ABSTRACT

Operations with tensors, or multiway arrays, have become increasingly prevalent in recent years. Traditionally, tensors are represented or decomposed as a sum of rank-1 outer products using either the CANDECOMP/PARAFAC (CP) or the Tucker models, or some variation thereof. Such decompositions are motivated by specific applications where the goal is to find an approximate such representation for a given multiway array. The specifics of the approximate representation (such as how many terms to use in the sum, orthogonality constraints, etc.) depend on the application.

In this paper, we explore an alternate representation of tensors which shows promise with respect to the tensor approximation problem. Reminiscent of matrix factorizations, we present a new factorization of a tensor as a product of tensors. To derive the new factorization, we define a closed multiplication operation between tensors. A major motivation for considering this new type of tensor multiplication is to devise new types of factorizations for tensors which can then be used in applications.

Specifically, this new multiplication allows us to introduce concepts such as tensor transpose, inverse, and identity, which lead to the notion of an orthogonal tensor. The multiplication also gives rise to a linear operator, and the null space of the resulting operator is identified. We extend the concept of outer products of vectors to outer products of matrices. All derivations are presented for third-order tensors. However, they can be easily extended to the order-p ($p > 3$) case. We conclude with an application in image deblurring.

☆ This work was supported in part by NSF grants DMS-0552577, DMS-0914957, and DMS-0914974.
* Corresponding author.
E-mail addresses: misha.kilmer@tufts.edu (M.E. Kilmer), carlam@math.jmu.edu (C.D. Martin).

0024-3795/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.laa.2010.09.020

1. Introduction

With the availability of cheap memory and advances in instrumentation and technology, it is now possible to collect and store more data for science, medical, and engineering applications than ever before. Often, this data is multidimensional in nature, as opposed to bi-dimensional: the information is stored in multiway arrays, known as tensors, as opposed to matrices. Applications involving operations with tensors include chemometrics [41], psychometrics [24], signal processing [9,27,39], computer vision [44–46], data mining [1,38], graph analysis [21], neuroscience [3,30,31], and many more. A common thread in such applications is the need to compress, sort, and/or otherwise manipulate the data by taking advantage of its multidimensional structure (see for example the recent article [34]). Collapsing multiway data to matrices and using standard linear algebra to answer questions about the data often has undesirable consequences.

In this paper the focus is on third-order tensors. However, our approach naturally generalizes to higher-order tensors in a recursive manner. Two well-known representations of third-order tensors are the CANDECOMP/PARAFAC (CP) [7,15] and Tucker3 [43] models. CP and Tucker3 are generally expressed as a sum of outer products of vectors, although in the literature, they are sometimes written using n-mode multiplication notation [22]. Each model can be considered an extension of the singular value decomposition (SVD) [12, p. 70] for matrices. In particular, one method for computing the Tucker3 decomposition is now commonly referred to as the higher-order SVD (HOSVD) from [26]. However, the overall theme in multiway data analysis is to build minimal approximations to a given tensor that satisfy the model and any additional constraints.

Our contribution is an alternative representation for tensors with the same ultimate goal in mind: building approximations to a given tensor. We think a bit 'outside the box' to give a representation of a tensor as the 'product' of two tensors which is reminiscent of the matrix factorization approach. This leads to a different generalization of the matrix SVD. We discuss how to use this generalization to derive low-rank tensor approximations. Furthermore, our framework allows other matrix factorizations to be extended to third-order tensors. Such higher-order extensions can then be used to give optimal representations in the Frobenius norm of tensors as a sum of so-called outer products of matrices.

Tensor decompositions or representations have been motivated by applications. As such, there are many representations of a tensor appearing in the literature, with no one representation being omnipresent in all applications. For a lengthy list of tensor representations and corresponding applications, see the recent review article on tensor-based approaches [22].

In this paper, we introduce a new tensor representation and compression algorithms based on a new tensor multiplication scheme. Hence, we offer new contributions to the class of tensor-based algorithms for compression. We emphasize that our contributions are not meant as a replacement for the many useful tensor representations presented in [22]. The tensor representation and algorithms here are orientation-specific which are useful for applications where the data has a fixed orientation, such as time series applications. Some examples include video compression where the third-order tensor contains two-dimensional images over time [19], handwritten digit identification [38], and image deblurring which is presented at the end of this paper. Since our representation is based on a fundamentally new tensor multiplication concept, we also hope to stimulate new research within the tensor community.

Our presentation is organized as follows. In Section 2, we describe the existing outer product representations most traditionally used in tensor representations and give some notation. In Section 3, we define a new type of multiplication between tensors and give corresponding notions of identity, inverse, and orthogonality. In Section 4, we give tensor–product decompositions based on these new definitions which resemble matrix factorizations and show how these lead to a natural low rank product decomposition of tensors. Section 5 illustrates the potential utility of our new representations on an application in image deblurring. We conclude with remarks on future work in Section 6.

2. Tensor background and notation

We use the accepted notation where an order-p tensor is indexed by p indices and can be represented as a multidimensional array of data [17]. That is, an order-p tensor, $\mathcal{A}$, can be written as

$$\mathcal{A} = (a_{i_1 i_2 \cdots i_p}) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}.$$

Thus, a matrix is considered a second-order tensor, and a vector is a first-order tensor. A third-order tensor can be pictured as a "cube" of data (see Fig. 2.1). While the orientation of third-order tensors is not unique, it is convenient to refer to its *slices*, i.e., the two-dimensional sections defined by holding two indices constant. We use the terms horizontal, lateral, and frontal slices defined in [22] to specify which two indices are held constant. Using MATLAB notation, $\mathcal{A}(k, :, :)$ corresponds to the kth horizontal slice, $\mathcal{A}(:, k, :)$ corresponds to the kth lateral slice, and $\mathcal{A}(:, :, k)$ corresponds the kth frontal slice. A *tube* of a third-order tensor is defined by holding the first two indices fixed and varying the third (see [22]). For example, using MATLAB notation, $\mathcal{A}(i, j, :)$ is the ijth tube of $\mathcal{A}$.

Throughout this paper, it is crucial to understand the orientation of a tensor. With that in mind, we restrict ourselves to third-order tensors and avoid messy subscripting wherever possible. Hence, we will mostly be referring to the frontal slices of a tensor, based on a given orientation.

If u is a length-m vector, v is a length-n vector, then $u \circ v$ is the *outer product* of u and v. The outer product gives a rank-1 matrix, whose (i, j)-entry is given by the scalar product $u_i v_j$. Similarly, the outer product $u \circ v \circ w$ yields a rank-1, third-order tensor with (i, j, k)-entry given by $u_i v_j w_k$. Likewise, an outer product of four vectors gives a rank-1, fourth-order tensor, etc.

The *tensor rank*, r, of an order-p tensor $\mathcal{A}$ is the minimum number of rank-1 tensors needed to express the tensor. For a third-order tensor, $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, this means we have the representation

$$\mathcal{A} = \sum_{i=1}^{r} \sigma_i (u^{(i)} \circ v^{(i)} \circ w^{(i)}), \tag{2.1}$$

where σ_i is a scaling constant. The scaling constants are simply the nonzero elements of an $r \times r \times r$ diagonal tensor $\Sigma = (\sigma_{ijk})$ (a tensor is *diagonal* if the only nonzeros occur in elements σ_{ijk} where $i = j = k$, see [22]). The vectors $u^{(i)}$, $v^{(i)}$, and $w^{(i)}$ are the ith columns from matrices $U \in \mathbb{R}^{n_1 \times r}$, $V \in \mathbb{R}^{n_2 \times r}$, $W \in \mathbb{R}^{n_3 \times r}$, respectively.

A decomposition of the form (2.1) is called a CANDECOMP–PARAFAC (CP) decomposition (CANonical DECOMPosition or PARAllel FACtors model) [7,15], whether or not r is known to be minimal. Note that the matrices U, V, W in (2.1) *are not constrained to be orthogonal*. Furthermore, an orthogonal decomposition of the form (2.1) may not exist [10]. There is no known closed-form solution to determine the rank r of a tensor *a priori*. Rank determination of a tensor is a widely-studied problem (see, for example [4,16,22,25,29]).

While some applications use the nonorthogonal decomposition (2.1), other applications need orthogonality of the matrices for better interpretation of the data [32,38,44–46]. Therefore, a more general form, called the Tucker3 decomposition [43] is often used to guarantee existence of an orthogonal decomposition as well as to better model certain data. The Tucker3 decomposition has also been called the higher-order SVD (HOSVD) [26], though the HOSVD actually refers to a method for computation [22]. However, Lathauwer et al. [26] shows that the HOSVD is a convincing extension of the matrix SVD. The HOSVD is guaranteed to exist and computes a Tucker3 decomposition directly. HOSVD first computes the SVDs of the matrices obtained by "flattening" the tensor in each dimension and then uses the results to assemble the so-called core tensor. The Tucker3 decomposition can be re-written as a CP decomposition, except that r will not typically correspond to the tensor rank.

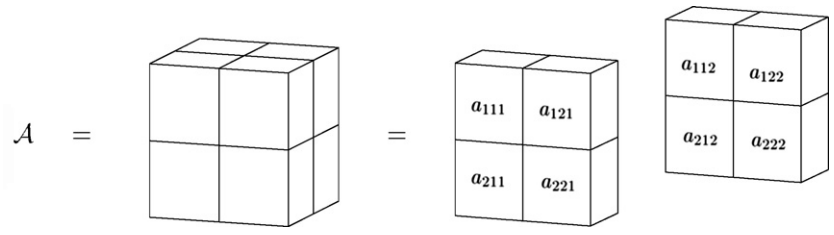

Fig. 2.1. Illustration of a $2 \times 2 \times 2$ tensor as a cube of data.

The CP and Tucker3 decompositions are analogous to the matrix SVD in that they describe the tensor as a sum of outer products of vectors. In geometric terms, the SVD decomposes a matrix into an outer product of vectors, which are one dimension less than a matrix. However, in the third-order case a vector has two fewer dimensions than a third-order tensor. Thus, one contribution of this paper is a decomposition of a third-order tensor that is an outer product of matrices (i.e., a decomposition into terms of only one dimension less). We pursue this idea further in Section 4.1.

We note that both the CP and Tucker3 representations can also be described in terms of *n-mode multiplication* [22] between a tensor and a matrix which is a way of expressing the vector outer-product sum as a matrix product involving a flattening of different stackings of the slices of the tensors Σ and $\mathcal{A}$. As our representations are based on a fixed orientation, we choose to avoid this notation.

Since we are working with third-order tensors, it is convenient to write (2.1) using Kruskal notation [23]. If $\mathcal{A} = \sum_{i=1}^{r} u^{(i)} \circ v^{(i)} \circ w^{(i)}$, is an $n_1 \times n_2 \times n_3$ tensor, then we may equivalently write $\mathcal{A} = [[U, V, W]]$, where the columns of U, V, W are $u^{(i)}$'s, $v^{(i)}$'s and $w^{(i)}$'s, respectively. It follows that U, V, W have r columns but U has n_1 rows, V has n_2 rows and W has n_3 rows.

In this paper, script notation is used to refer to tensors. Capital non-script letters are used to refer to matrices and lower case letters refer to vectors. Entries in vectors are indexed by subscripts. We use $\text{diag}(v_1, \ldots, v_n)$ to denote the $n \times n$ diagonal matrix with entries $v_1, \ldots, v_n$. Similarly, the notation $\text{diag}(D_1, \ldots, D_k)$, for k, $n_1 \times n_2$ matrices D_i, refers to a block diagonal matrix of size $kn_1 \times kn_2$ with $n_1 \times n_2$ blocks.

2.1. Approximate tensor factorizations

We adopt the definition of the Frobenius norm of a tensor used in the literature:

Definition 2.1. Suppose $\mathcal{A} = (a_{ijk})$ is size $n_1 \times n_2 \times n_3$. Then

$$\|\mathcal{A}\|_F = \sqrt{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3} a_{ijk}^2}.$$

One of the fundamental problems in applications is finding a CP or Tucker3 approximation, $\widetilde{\mathcal{A}}$, to a given tensor $\mathcal{A}$ which is optimal in the sense that

$$\min \|\mathcal{A} - \widetilde{\mathcal{A}}\|_F, \quad \text{subject to vector constraints} \tag{2.2}$$

is solved, possibly subject to some constraints on $\widetilde{\mathcal{A}}$ as well.

Computing the rank of a tensor as in (2.1) is not well-posed. One idea that has been explored for determining good low-rank approximations to $\mathcal{A}$, based on this fact, is to successively subtract best rank-1 approximations from $\mathcal{A}$. Unfortunately, this process does not necessarily lead to tensors that have subsequently lower rank [42]. It may be that the best rank-k approximation does not exist [22,35,40]. However, the best rank-1 problem is solveable. Algorithms for finding this (iteratively) can be found in [20,28,47].

If an estimate, r_k of the rank, r, is available, one could use the CP model (2.1) where $\widetilde{\mathcal{A}}$ in (2.2) is found using $r_k \approx r$. Computing the best rank-(k_1, k_2, k_3) approximation (known as the *multilinear rank*) is well-posed (see [25,26]).

In some cases, the vectors in the CP and Tucker3 decompositions are constrained to be nonnegative [6,11] or orthogonal [32] given the physical nature of the problem. See Ref. [22] for a more complete list of constrained tensor decompositions and corresponding algorithms.

In Section 4, however, we give a method for subtracting low rank approximations from $\mathcal{A}$ based on a new type of factorization into a tensor SVD, which gives a handle on the quality of the approximation after each successive step. Our strategy, loosely outlined, is as follows:

- Find $\widetilde{\mathcal{A}} = \arg\min_{\mathcal{A} \in M} \|\mathcal{A} - \widetilde{\mathcal{A}}\|_F$, where M describes a special class of tensors that can be written as a "product" of tensors of appropriate dimension.

- Compute a low-rank approximation to $\widetilde{\mathcal{A}}$.
- Repeat, as necessary, on $\mathcal{A} - \widetilde{\mathcal{A}}$.

First, however, we need to introduce the type of multiplication that will give rise to such product-based factorizations.

3. New tensor operators

One major contribution of this paper is an alternative tensor representation based on a product of two tensors, which we will call the t-product.[1] In this section, we define a new notion of multiplication between tensors and present other properties that follow from our new definition. The t-product operator was initially motivated by desiring a closed operation that preserves the order of a tensor. We begin by showing where current multiplication strategies fall short in this regard. Then we introduce the the t-product operation and corresponding group-theoretic and linear-algebraic properties.

3.1. Current tensor multiplication strategies

Multiplication between tensors and matrices has been defined using the n-mode product [2,26,17]. We will not go into detail here, except to describe the 1-mode product, which will reappear a few times throughout, due to the fixed orientation in which we are working. If $\mathcal{A}$ is an $n_1 \times n_2 \times n_3$ tensor, then the 1-mode product of $\mathcal{A}$ with $n_2 \times n_1$ matrix U, is the $n_2 \times n_2 \times n_3$ tensor that results from left multiplying each frontal slice of $\mathcal{A}$ with U.

There are several ways to multiply tensors, but the most common method is the *contracted product*. The name "contracted product" can be a little misleading: indeed, the contracted product of an $\ell \times n_2 \times n_3$ tensor and an $\ell \times m_2 \times m_3$ tensor in the first mode is an $n_2 \times n_3 \times m_2 \times m_3$ tensor. For example, if $\mathcal{A}$ is $n_1 \times n_2 \times n_3$ and $\mathcal{B}$ is $n_1 \times m_2 \times m_3$, then the contracted product of $\mathcal{A}$ and $\mathcal{B}$ in the first "mode" or "dimension" is $n_2 \times n_3 \times m_2 \times m_3$. However, the contracted product of an $n_1 \times n_2 \times n_3$ tensor and a $n_1 \times n_2 \times m_3$ tensor in the first two modes results in an $n_3 \times m_3$ tensor (matrix). Notably, the contracted product does not preserve the order of a tensor which suggests that it is perhaps not ideal for helping to generalize other concepts of linear algebra for tensors.

In summary, the order of the resulting tensor depends on the modes where the multiplication takes place. We refer the reader to the explanation in [2] for details. We now introduce a new definition of multiplication between tensors that preserves order. For example, the product of an $n \times n \times n$ tensor with another of the same dimension will yield an $n \times n \times n$ tensor. We start by giving some notation that will be useful in deriving the concept of multiplication between tensors.

3.2. Notation

We use circulant matrices extensively in our new definitions. If

$$v = \begin{bmatrix} v_0 & v_1 & v_2 & v_3 \end{bmatrix}^T$$

then

$$\operatorname{circ}(v) = \begin{bmatrix} v_0 & v_3 & v_2 & v_1 \\ v_1 & v_0 & v_3 & v_2 \\ v_2 & v_1 & v_0 & v_3 \\ v_3 & v_2 & v_1 & v_0 \end{bmatrix}$$

is a circulant matrix. Note that all the matrix entries are defined once the first column is specified. Therefore, we adopt the convention that $\operatorname{circ}(v)$ refers to the circulant matrix obtained with the vector v as the first column.

[1] This is to distinguish it from the notion of "tensor product", which often is understood to refer to the Kronecker product of two matrices.

Circulant matrices can be diagonalized with the normalized discrete Fourier transform (DFT) matrix [12, p. 202], which is unitary. In particular, if v is $n \times 1$, F_n is the $n \times n$ DFT matrix, and F_n^* is its conjugate transpose, then

$$F_n \operatorname{circ}(v) F_n^*$$

is diagonal. The following, well-known, simple fact [8] is used to compute this diagonal using the fast Fourier transform (FFT):

Fact 1. The diagonal of $F_n \operatorname{circ}(v) F_n^* = \operatorname{fft}(v)$, where $\operatorname{fft}(v)$ is the result of applying the fast Fourier transform to v.

It is possible to create a block circulant matrix from the slices of a tensor. For this paper, we will always assume the block circulant is created from the frontal slices (if we wished another ordering, we would first permute the tensor to achieve it), and thus there should be no ambiguity with the following notation. For example, if $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ with $n_1 \times n_2$ frontal slices $A_1, \ldots, A_{n_3}$ then

$$\operatorname{circ}(\mathcal{A}) = \begin{bmatrix} A_1 & A_{n_3} & A_{n_3-1} & \ldots & A_2 \\ A_2 & A_1 & A_{n_3} & \ldots & A_3 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ A_{n_3} & A_{n_3-1} & \ddots & A_2 & A_1 \end{bmatrix},$$

where $A_i = \mathcal{A}(:, :, i)$ for $i = 1, \ldots, n_3$.

Similarly, we will anchor the MatVec command to the frontal slices of the tensor. $\operatorname{MatVec}(\mathcal{A})$ takes an $n_1 \times n_2 \times n_3$ tensor and returns a block $n_1 n_3 \times n_2$ matrix

$$\operatorname{MatVec}(\mathcal{A}) = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_{n_3} \end{bmatrix}.$$

The operation that takes $\operatorname{MatVec}(\mathcal{A})$ back to tensor form is the fold command:

$$\operatorname{fold}(\operatorname{MatVec}(\mathcal{A})) = \mathcal{A}.$$

Just as circulant matrices can be diagonalized by the DFT, block-circulant matrices can be block-diagonalized. Suppose $\mathcal{A}$ is $n_1 \times n_2 \times n_3$ and F_{n_3} is the $n_3 \times n_3$ DFT matrix. Then

$$(F_{n_3} \otimes I_{n_1}) \cdot \operatorname{circ}(\operatorname{MatVec}(\mathcal{A})) \cdot (F_{n_3}^* \otimes I_{n_2}) = \begin{bmatrix} D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_{n_3} \end{bmatrix}, \tag{3.1}$$

where "$\otimes$" denotes the Kronecker product and F^* denotes the conjugate transpose of F and "$\cdot$" means standard matrix product. Note that each D_i could be dense and furthermore most will be complex unless certain symmetry conditions hold.

To compute the product in the preceding paragraph, assuming n_3 is a power of 2, can be done in $O(n_1 n_2 n_3 \log_2(n_3))$ flops using the FFT and Fact 1. Indeed, using stride permutations and Fact 1, it is straightforward to show that there is no need to lay out the data in order to compute the matrices D_i. Indeed, we have the following.

Fact 2. The D_i are the frontal slices of the tensor $\mathcal{D}$, where $\mathcal{D}$ is computed by applying FFT's along each tube of $\mathcal{A}$.

3.3. New tensor multiplication

In this section we define a new type of multiplication between tensors, called the t-product, and explore some of the important theoretical and practical resulting properties.

Definition 3.1. Let $\mathcal{A}$ be $n_1 \times n_2 \times n_3$ and $\mathcal{B}$ be $n_2 \times \ell \times n_3$. Then the t-product $\mathcal{A}*\mathcal{B}$ is the $n_1 \times \ell \times n_3$ tensor

$$\mathcal{A}*\mathcal{B} = \text{fold}(\text{circ}(\mathcal{A})) \cdot \text{MatVec}(\mathcal{B})).$$

Example 3.2. Suppose $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times 3}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times \ell \times 3}$. Then

$$\mathcal{A}*\mathcal{B} = \text{fold}\left(\begin{bmatrix} A_1 & A_3 & A_2 \\ A_2 & A_1 & A_3 \\ A_3 & A_2 & A_1 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}\right) \in \mathbb{R}^{n_1 \times \ell \times 3}.$$

If the tensors are sparse, we may choose to compute this product as it is written. If the tensors are dense, naively computing the t-product would cost $O(n_1 n_2 n_3^2 \ell)$ flops. However, since circ(MatVec($\mathcal{A}$)) can be block diagonalized, we can choose to compute this product as

$$(F_{n_3}^* \otimes I_{n_1})((F_{n_3} \otimes I_{n_1}) \cdot \text{circ}(\text{MatVec}(\mathcal{A})) \cdot (F_{n_3}^* \otimes I_{n_2}))(F_{n_3} \otimes I_{n_2})\text{MatVec}(\mathcal{B}).$$

It is readily shown that $(F_{n_3} \otimes I_{n_2})\text{MatVec}(\mathcal{B})$ can be computed in $O(\ell n_2 n_3 \log_2(n_3))$ flops by applying FFTs along the tubes of $\mathcal{B}$: we call the result $\widetilde{\mathcal{B}}$. If we take the FFT of each tube of $\mathcal{A}$, using Fact 2, we obtain $\mathcal{D}$. Thus, it remains to multiply each frontal slice of $\mathcal{D}$ with each frontal slice of $\widetilde{\mathcal{B}}$, then take an inverse FFT along the tubes of the result. We arrive at the following fact regarding this multiplication.

Fact 3. The t-product in Definition 3.1 can be computed in at most $O(n_1 n_2 \ell n_3)$ flops by making use of the FFT along mode 3.

If n_3 is not a power of two, we may still employ FFTs in the multiplication by noting that the block circulant matrix can be embedded in a larger block circulant matrix where the number of blocks in a block row can be increased to the next largest power of two greater than $2n_3 - 1$ by the addition of zero blocks and repetition of previous blocks in an appropriate fashion. Likewise, once $\mathcal{B}$ is unfolded, it can be conformally extended by zero blocks. The product is computed using FFTs, and the result is then truncated appropriately. This is a commonly used trick in the literature (see, for example [33]) for fast multiplication with Toeplitz or block Toeplitz matrices by embedding them in larger, block circulant circulant block matrices, and will not be described further here.

Now we discuss some group-theoretical properties of the t-product. First, the t-product is associative, as the next lemma shows.

Lemma 3.3. $\mathcal{A}*(\mathcal{B}*\mathcal{C}) = (\mathcal{A}*\mathcal{B})*\mathcal{C}$.

Proof. The proof follows naturally from the definition of $*$ and the fact that matrix–matrix multiplication is associative. □

Definition 3.4. The $n \times n \times \ell$ identity tensor $\mathcal{I}_{nn\ell}$ is the tensor whose frontal slice is the $n \times n$ identity matrix, and whose other frontal slices are all zeros.

It is clear that $\mathcal{A}*\mathcal{I} = \mathcal{A}$ and $\mathcal{I}*\mathcal{A} = \mathcal{A}$ given the appropriate dimensions.

For an $n \times n \times \ell$ tensor, an inverse exists if it satisfies the following:

Definition 3.5. An $n \times n \times \ell$ tensor $\mathcal{A}$ has an inverse $\mathcal{B}$ provided that

$$\mathcal{A}*\mathcal{B} = \mathcal{I}_{nn\ell}, \quad \text{and} \quad \mathcal{B}*\mathcal{A} = \mathcal{I}_{nn\ell}.$$

From Definitions 3.1, 3.4, 3.5 and Lemma 3.3, we have the following lemma.

Lemma 3.6. *The set of all invertible $n \times n \times n$ tensors forms a group under the $*$ operation.*

It is also true that the set of invertible $n \times n \times n$ tensors forms a ring under standard tensor addition (component-wise addition) and the t-product. Furthermore, as we show in the next section, the set of all invertible $n \times n \times n$ tensors is non-empty.

3.4. Linear operators, rank, and null space

We also can define linear transformations around the t-product.

Lemma 3.7. *If $T(\mathcal{X}) = \mathcal{A}*\mathcal{X}$ where $\mathcal{A}$ is a real $n_1 \times m \times n_3$ and $\mathcal{X}$ is a real $m \times n_2 \times n_3$ tensor, then $T : \mathbb{R}^{m\times n_2\times n_3} \to \mathbb{R}^{n_1\times n_2\times n_3}$ is linear.*

Proof. Follows directly from the definition and the linearity of matrix–matrix products. □

We note that Braman [5] was able to show that our t-product results in a linear operator in a special case when $n_2 = 1$.

In particular, since the mode-1 product can be represented using this new notation, mode-1 multiplication defines a linear transformation. This is in contrast to the interpretation (see [22, p. 6]) that a mode-1 multiplication defines a change of basis when the tensor defines a *multilinear* operator.

Since the t-product defines a linear operator, it makes sense to explore invertibility and the null space, which is more easily accomplished via the following result.

Theorem 3.8. *Let $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2\times \ell\times n_3}$ be rank r_A and r_B tensors, respectively, defined by*

$$\mathcal{A} = \sum_{i=1}^{r_A} u^{(i)} \circ v^{(i)} \circ w^{(i)}, \quad \mathcal{B} = \sum_{j=1}^{r_B} x^{(j)} \circ y^{(j)} \circ z^{(j)}.$$

Let $t^{(i,j)}$ denote the vector $\mathrm{circ}(w^{(i)})z^{(j)}$. *Define scalars $d_{ij} = (v^{(i)})^T x^{(j)}$. Then*

$$\mathcal{A}*\mathcal{B} = \sum_{i=1}^{r_A}\sum_{j=1}^{r_B} d_{ij}(u^{(i)} \circ y^{(j)} \circ t^{(i,j)}). \tag{3.2}$$

Proof. Using the definition to lay out the tensor–product as the product of two matrices,

$$\sum_{i=1}^{r_A} \mathrm{circ}(w^{(i)}) \otimes u^{(i)}(v^{(i)})^T \sum_{j=1}^{r_B} z^{(j)} \otimes x^{(j)}(y^{(j)})^T = \sum_{i=1}^{r_A}\sum_{j=1}^{r_B} d_{ij}(\mathrm{circ}(w^{(i)})z^{(j)}) \otimes u^{(i)}(y^{(j)})^T,$$

where the last equality comes from properties of Kronecker products. The result follows upon applying the MatVec operation to the matrix on the right. □

In the following, we use $\mathcal{T}$ to denote the tensor with ij-tubes, $t^{(i,j)}$, for simplicity. Now we discuss rank and null space.

Corollary 3.9. *Let T denote the $r_A \times r_B$ matrix formed by taking the norms (any valid vector norm) of the tubes $t^{(i,j)}$ of $\mathcal{T}$, and let Δ be the matrix with entries d_{ij}.*

Then $rank(\mathcal{A}*\mathcal{B}) = nnz(\Delta \odot T) \leqslant r_A r_B$, where nnz means number of nonzeros and $\odot$ stands for Hadamard product. Furthermore, $\mathcal{B} \in null(\mathcal{A})$ if $\|\Delta \odot T\| = 0$.

Proof. Follows by noting that an entry in the matrix T will be zero precisely when $z^{(j)}$ is in the null space of $\mathrm{circ}(w^{(i)})$. □

In the remainder of the paper, we use the notation $\hat{v}$ to denote the vector of Fourier coefficients of a vector v. Note that entry i,j in T will be zero precisely when the vector $\hat{w}^{(i)} \odot \hat{z}^{(j)}$ is the zero vector. Therefore,

Corollary 3.10. *Fix i, and assume that $\text{circ}(w^{(i)})z^{(i)} = s^{(i)} \neq 0$. Then, $\text{circ}(w^{(i)})z^{(j)} = 0$ for some $j \neq i$ is possible only if $\hat{s}^{(i)}$ has at least 1 zero entry.*

Next, we move on to invertibility.

Theorem 3.11. *Let $\mathcal{A} = \sum_{i=1}^{r} u^{(i)} \circ v^{(i)} \circ w^{(i)}$ be a representation of an $n_1 \times m \times n_3$ tensor.[2] Then $\mathcal{A}$ has an $m \times n_1 \times n_3$ right inverse, $\mathcal{A}^\dagger$, such that $\mathcal{A} * \mathcal{A}^\dagger = \mathcal{I}_{n_1 n_1 n_3}$ defined by*

$$\mathcal{A}^\dagger = \sum_{i=1}^{r} x^{(i)} \circ y^{(i)} \circ z^{(i)}$$

provided that $UY^T = I_{n_1}$, $V^T X = I_r$ are solvable and that each $w^{(i)}$ has no zero Fourier coefficients, so that $\hat{z}^{(i)} = 1./\hat{w}^{(i)}$. In particular, $\mathcal{A}^{-1} \equiv \mathcal{A}^\dagger$ when U, V are square and full rank.

Proof. If $V^T X = I_r$ and $\text{circ}(w^{(i)})z^{(j)} = e_1$, then by Theorem 3.8, the product $\mathcal{A} * \mathcal{A}^\dagger = \left(\sum_{i=1}^{r} u^{(i)} \circ y^{(i)}\right) \circ e_1 = \mathcal{I}_{n_1 \times n_2 \times n_3}$ precisely when $UY^T = I_{n_1 \times n_2}$. In order for $\text{circ}(w^{(i)})z^{(i)} = e_1$, by taking the Fourier transform of each side we need $\hat{z}^{(i)} = 1./\hat{w}^{(i)}$. □

Thus, for existence of a (right) inverse, this means we need $r \geqslant n_1$ and U to have full rank, and $m \geqslant r$ with V full rank.

For applications purposes (see Section 5), it is also convenient to define a right pseudoinverse in the case $r < n_1$.

Definition 3.12. If $\mathcal{A} = \sum_{i=1}^{r} u^{(i)} \circ v^{(i)} \circ w^{(i)}$ is $m \times n_2 \times n_3$ then if $\hat{z}^{(i)}$ has no 0 entries:

$$\mathcal{A}^{\dagger\dagger} = \sum_{i=1}^{r} x^{(i)} \circ y^{(i)} \circ z^{(i)} \in \mathbb{R}^{n_1 \times m \times n_3}$$

with $Y^T = U^\dagger$, $V^T X = I_r$ and $\hat{z}^{(i)} = 1./\hat{w}^{(i)}$.

A similar definition for a left pseudoinverse is also possible.

Now it makes sense to consider whether or not a tensor of rank r can be factored as a product of two tensors of rank no greater than r. This is straightforward with appropriate definition of the "free" parameters in Theorem 3.8 (i.e., we can ensure it if $V^T X = D$ is an invertible diagonal matrix, for instance; however if this is not the case, it still might be possible to do, depending on the Fourier coefficients of the 3rd terms in the outer product representation). The factorization is not unique, although some terms are specified.

For completeness, we note that we have a change-of-basis type of result[3]:

Theorem 3.13. *Given $\mathcal{C} = \sum_{j=1}^{r} p^{(j)} \circ q^{(j)} \circ s^{(j)}$. Let $P = [p^{(1)}, \ldots, p^{(r)}] = UE$, where U is $n_1 \times k_1$ with k_1 linearly independent columns. Then*

$$\mathcal{C} = \mathcal{A} * \mathcal{B}, \quad \mathcal{A} = \sum_{i=1}^{k_1} u^{(i)} \circ v^{(i)} \circ e_1, \quad \mathcal{B} = \sum_{j=1}^{r} x^{(j)} \circ q^{(j)} \circ s^{(j)},$$

as long as $V^T X = E$, with e_1 the first column of the $n_3 \times n_3$ identity matrix.

[2] We have not assumed that r is minimal here.

[3] Compare to p. 6 of Ref. [22].

3.5. Transpose, orthogonality, range

Armed with the definition of t-product, several interesting facts now arise (dimensions relative to Lemma 3.7):

- If we take $m = 1$, we arrive at something that is akin to the outer-product of two vectors. The outer-product of two vectors gives a matrix. Here, the "outer-product" of two matrices (i.e., $m = 1$ but $n_1 > 1,\ n_2 > 1,\ n_3 > 1$) gives an $n_1 \times n_2 \times n_3$ tensor. We show based on the remainder of the definitions in this section, that we can construct an optimal (in the Frobenius norm) factorization of a tensor into a sum of outer products of matrices, given our fixed orientation.
- In linear algebra, it is quite common to think of the matrix–matrix product AB as A acting on each column of the matrix, and each column is a vector: $AB = [Ab_1, \ldots, Ab_n]$. Similarly, if $\mathcal{C} = \mathcal{A}*\mathcal{B}$, each lateral slice of $\mathcal{C}$ (a matrix) is obtained by $\mathcal{A}$ acting on a lateral slice of $\mathcal{B}$ (also a matrix) and so we have $\mathcal{C}(:, i, :) = \mathcal{A}*\mathcal{B}(:, i, :)$.
- If we take $n_1 = 1 = n_2$, but $m > 1,\ n_3 > 1$, the result is a single tube, which can be oriented as a vector. Thus, an "inside-product" (indeed, in a forthcoming work, we show that this satisfies properties of an inner product) of two matrices, appropriately oriented as tensors, results in a vector.

Our goal in this section is to build on the t-product definition, to try to take advantage of some of the observations above. First, we need a few more definitions.

With the definition of a transpose operation for tensors, we will be able to write our previous approximation in terms of products of tensors.

Definition 3.14. If $\mathcal{A}$ is $n_1 \times n_2 \times n_3$, then $\mathcal{A}^T$ is the $n_2 \times n_1 \times n_3$ tensor obtained by transposing each of the frontal slices and then reversing the order of transposed frontal slices 2 through n_3.

Example 3.15. If $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times 4}$ and its frontal slices are given by the $n_1 \times n_2$ matrices $A_1,\ A_2,\ A_3,\ A_4$, then

$$\mathcal{A}^T = \text{fold}\left(\begin{bmatrix} A_1^T \\ A_4^T \\ A_3^T \\ A_2^T \end{bmatrix}\right).$$

The tensor transpose has the same property as the matrix transpose.

Lemma 3.16. *Suppose $\mathcal{A},\ \mathcal{B}$ are two tensors such that $\mathcal{A}*\mathcal{B}$ and $\mathcal{B}^T*\mathcal{A}^T$ is defined. Then $(\mathcal{A}*\mathcal{B})^T = \mathcal{B}^T*\mathcal{A}^T$.*

Proof. Follows directly from Definitions 3.1 and 3.14. □

For completeness we define permutation tensors.

Definition 3.17. A permutation tensor is an $n \times n \times \ell$ tensor $\mathcal{P} = (p_{ijk})$ with exactly n entries of unity, such that if $p_{ijk} = 1$, it is the only non-zero entry in row i, column j, and tube k.

We are now ready to define orthogonality for tensors, from which it follows that the identity tensor and permutation tensors are orthogonal.

Definition 3.18. An $n \times n \times \ell$ real-valued tensor $\mathcal{Q}$ is orthogonal if $\mathcal{Q}^T*\mathcal{Q} = \mathcal{Q}*\mathcal{Q}^T = \mathcal{I}$.

We can also define a notion of partial orthogonality, similar to saying that a tall, thin matrix has orthogonal columns. In this case if $\mathcal{Q}$ is $p \times q \times n$ and **partially orthogonal**, we mean $\mathcal{Q}^T*\mathcal{Q}$ is well

defined and equal to the a $q \times q \times n$ identity. Note that if $\mathcal{Q}$ is an orthogonal tensor, then *it does not follow that each frontal slice of $\mathcal{Q}$ is necessarily orthogonal.*

Another nice feature of orthogonal (similarly, partially tensors) is that they preserve the Frobenius norm:

Lemma 3.19. *If $\mathcal{Q}$ is an orthogonal tensor,*

$$\|\mathcal{Q}*\mathcal{A}\|_F = \|\mathcal{A}\|_F.$$

Proof. From Definitions 3.1, 3.14, and 2.1, it follows that

$$\|\mathcal{A}\|_F^2 = \text{trace}((\mathcal{A}*\mathcal{A}^T)_{(:,:,1)}) = \text{trace}((\mathcal{A}^T*\mathcal{A})_{(:,:,1)}),$$

where $(\mathcal{A}*\mathcal{A}^T)_{(:,:,1)}$ is the frontal slice of $\mathcal{A}*\mathcal{A}^T$ and $(\mathcal{A}^T*\mathcal{A})_{(:,:,1)}$ is the frontal slice of $\mathcal{A}^T*\mathcal{A}$. Therefore,

$$\begin{aligned}\|\mathcal{Q}*\mathcal{A}\|_F^2 &= \text{trace}([(\mathcal{Q}*\mathcal{A})^T*(\mathcal{Q}*\mathcal{A})]_{(:,:,1)}) \\ &= \text{trace}([\mathcal{A}^T*\mathcal{Q}^T*\mathcal{Q}*\mathcal{A}]_{(:,:,1)}) \\ &= \|\mathcal{A}\|_F^2. \quad \square\end{aligned}$$

Note that if the tensor is two-dimensional (i.e., $n_3 = 1$, so the tensor is a matrix), Definitions 2.1, 3.1, 3.4, 3.5, 3.17, and 3.18 are consistent with standard matrix algebra operations and terminology.

We are finally in a position to consider tensor factorizations that are analagous to the matrix SVD and matrix QR.

4. New product decompositions of tensors

We say a tensor is "f-diagonal" if each *frontal slice* is diagonal. Likewise, a tensor is f-upper triangular or f-lower triangular if each *frontal slice* is upper or lower triangular, respectively.

Theorem 4.1 (T-SVD)**.** *Let $\mathcal{A}$ be an $n_1 \times n_2 \times n_3$ real-valued tensor. Then $\mathcal{A}$ can be factored as*

$$\mathcal{A} = \mathcal{U}*\mathcal{S}*\mathcal{V}^T, \tag{4.1}$$

where $\mathcal{U}$, $\mathcal{V}$ are orthogonal $n_1 \times n_1 \times n_3$ and $n_2 \times n_2 \times n_3$, respectively, and $\mathcal{S}$ is a $n_1 \times n_2 \times n_3$ f-diagonal tensor. The factorization (4.1) is called the T-SVD (i.e., tensor SVD).

Proof. The proof is by construction. First, we transform $\text{circ}(\mathcal{A})$ into the Fourier domain as in (3.1). Next, we compute the SVD of each D_i as $D_i = U_i \Sigma_i V_i^T$. Then

$$\begin{bmatrix} D_1 & & \\ & \ddots & \\ & & D_{n_3} \end{bmatrix} = \begin{bmatrix} U_1 & & \\ & \ddots & \\ & & U_{n_3} \end{bmatrix} \begin{bmatrix} \Sigma_1 & & \\ & \ddots & \\ & & \Sigma_{n_3} \end{bmatrix} \begin{bmatrix} V_1^T & & \\ & \ddots & \\ & & V_{n_3}^T \end{bmatrix}. \tag{4.2}$$

We apply $(F_{n_3}^* \otimes I)$ to the left and $(F_{n_3} \otimes I)$ to the right of each of the block diagonal matrices in (4.2). Observing that in each of the three cases, the resulting triple product results in a block circulant matrix, we define $\text{MatVec}(\mathcal{U})$, $\text{MatVec}(\mathcal{S})$, $\text{MatVec}(\mathcal{V}^T)$ as the first block columns of each of the respective block-circulant matrices, and fold the results. This gives a decomposition of the form $\mathcal{U}*\mathcal{S}*\mathcal{V}^T$.

It remains to show that $\mathcal{U}$ and $\mathcal{V}$ are orthogonal. However, this is easily proved by forming the necessary products (e.g. $\mathcal{U}^T*\mathcal{U}$) and using the same forward, backward matrix transformation to the Fourier domain as was used to compute the factorization, and the proof is complete. $\square$

We note that this particular diagonalization was achieved using the standard decreasing ordering for the singular values of each D_i. If a different ordering is used, a different diagonalization would be

achieved, which would be equivalent up to permutation (in the Fourier domain) with the T-SVD given here.

Assuming (4.1) and using Lemma 3.19 we have that $\|\mathcal{A}\|_F = \|\mathcal{S}\|_F$. We will make use of this fact in the next section in devising approximation strategies based on (4.1). The T-SVD can be computed using the fast Fourier transform utilizing Fact 1 from Section 3.2. One version of MATLAB pseudocode is provided below.

Algorithm T-SVD

Input: $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$
$\mathcal{D} = \text{fft}(\mathcal{A}, [], 3)$;
for $i = 1, \ldots, n_3$
$[U, S, V] = \text{svd}(\mathcal{D}(:,:,i))$;
$\mathcal{U}(:,:,i) = u$; $\mathcal{V}(:,:,i) = v$; $\mathcal{S}(:,:,i) = s$
$\mathcal{U} = \text{ifft}(\mathcal{U}, [], 3)$; $\mathcal{V} = \text{ifft}(\mathcal{V}, [], 3)$; $\mathcal{S} = \text{ifft}(\mathcal{S}, [], 3)$;

Note that if $\mathcal{A}$ is real, (4.1) is composed of real tensors even though the proof of Theorem 4.1 involves computations over the complex field. The complex computations result when computing the D_i matrices in (4.2). In particular, these D_i matrices will be complex unless there are very specific symmetry conditions imposed on the original tensor.

The T-SVD and the SVD of the matrix $\sum_{i=1}^{n_3} \mathcal{A}(:,:,i)$ are related as the following Lemma shows.

Lemma 4.2. *Suppose the T-SVD of $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is given by $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$. Then*

$$\sum_{k=1}^{n_3} \mathcal{A}(:,:,k) = \left(\sum_{k=1}^{n_3} \mathcal{U}(:,:,k)\right)\left(\sum_{k=1}^{n_3} \mathcal{S}(:,:,k)\right)\left(\sum_{k=1}^{n_3} \mathcal{V}(:,:,k)^T\right), \tag{4.3}$$

Furthermore, (4.3) gives an SVD for $\sum \mathcal{A}(:,:,k)$ in the sense that $\sum \mathcal{U}(:,:,k)$, $\sum \mathcal{V}(:,:,k)$ are orthogonal and $\sum \mathcal{S}(:,:,k)$ is diagonal.

Proof. Clearly $\sum \mathcal{S}(:,:,k)$ is a diagonal matrix (the entries can be made positive by an appropriate scaling). Now, all that remains to show is that if $\mathcal{U}$ is an orthogonal tensor, then $\sum \mathcal{U}(:,:,k)$ is an orthogonal matrix (the proof for $\sum \mathcal{V}(:,:,k)$ follows similarly).

Suppose $\mathcal{U}$ is orthogonal. Then we have $\mathcal{U} * \mathcal{U}^T = \mathcal{I}_{n_1 n_1 n_3}$ which means, by Definition 3.1 that

$$\sum_{k=1}^{n_3} (\mathcal{U}(:,:,k)\mathcal{U}(:,:,k)^T) = I_{n_1} \quad \text{and} \quad \sum_{i \neq j} (\mathcal{U}(:,:,i)\mathcal{U}(:,:,j)^T) = \mathbf{0}_{n_1}. \tag{4.4}$$

Eq. (4.4) means that $\left(\sum_{k=1}^{n_3} \mathcal{U}(:,:,k)\right)\left(\sum_{k=1}^{n_1} \mathcal{U}(:,:,k)\right)^T = I_{n_1}$ which completes the proof. □

We now have an interesting way to describe the range of our linear operator, based on the SVD and bullet points at the beginning of this section, which is analogous to the matrix case. We can say $\mathcal{B}$ is in the range of $T(\mathcal{X}) = \mathcal{A} * \mathcal{X}$ if $\mathcal{B}(:,j,:)$ is of the form $\sum_{i=1}^{n} \mathcal{U}(:,i,:) * c(i,i,:)$ for each j. Note this is not quite a linear combination in the sense that the $c(i,i,:)$ are not scalars, but they do represent the "inside products" $(\mathcal{S}(i,i,:) * \mathcal{V}(:,i,:)^T) * \mathcal{X}(:,i,:)$ for some $\mathcal{X}$.

Other matrix factorization ideas can be extended to third-order tensors in a similar fashion as the T-SVD. For example, we can compute a QR type decomposition $\mathcal{A} = \mathcal{Q} * \mathcal{R}$ [19] where $\mathcal{Q}$ is an orthogonal tensor and $\mathcal{R}$ is f-upper triangular. We call this a T-QR factorization. Such a decomposition might be preferred when data is being added to each frontal slice of the tensor, because QR-updating strategies can be employed (in the Fourier domain). Note that the T-SVD and T-QR decompositions can be done in "reduced" form analogous to matrices when $\mathcal{D}(:,:,i)$ is rectangular. For example, if $\mathcal{D}(:,:,i)$ is $n_1 \times n_2$, $n_1 > n_2$, we can compute its reduced SVD, rather than the full SVD, in which case each matrix U is no longer orthogonal but has $n_2 < n_1$ orthonormal columns. As a result, $\mathcal{U}$ will be partially orthogonal $n_1 \times n_2 \times n_3$, rather than orthogonal, and $\mathcal{S}$ will be $n_2 \times n_2 \times n_3$.

Finally, we note again that the factorizations are orientation dependent: i.e., rotating the tensor gives a different factorization. On the other hand, applying permutation tensors to $\mathcal{A}$ before computing the T-SVD does not affect the entries in $\mathcal{S}$, and affects the right or left singular tensor through this permutation.

4.1. Approximation strategies based on products of tensors

In [19] we present a compression strategy based on Lemma 4.2. The compression is based on the assumption that the terms $\|S(i,i,:)\|_F^2$ decay rather quickly. We do not pursue this idea further here, but rather present a compression strategy based on the following. If the T-SVD of $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$ is given by $\mathcal{A} = \mathcal{U}*\mathcal{S}*\mathcal{V}^T$, then it is easy to show that

$$\mathcal{A} = \sum_{i=1}^{\min(n_1,n_2)} \mathcal{U}(:,i,:)*\mathcal{S}(i,i,:)*\mathcal{V}(:,i,:)^T. \tag{4.5}$$

Thus, $\mathcal{A}$ is written as a finite sum of outer products *of matrices*. A particularly nice feature of the T-SVD is that it gives a way to find an optimal approximation of a tensor as a sum of $k < \min(n_1,n_2)$ of the matrix outer products in (4.5).

Theorem 4.3. *Let the T-SVD of $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$ be given by $\mathcal{A} = \mathcal{U}*\mathcal{S}*\mathcal{V}^T$ and for $k < \min(n_1,n_2)$ define*

$$\mathcal{A}_k = \sum_{i=1}^{k} \mathcal{U}(:,i,:)*\mathcal{S}(i,i,:)*\mathcal{V}(:,i,:)^T.$$

Then $\mathcal{A}_k = \arg\min_{\widetilde{\mathcal{A}}\in M} \|\mathcal{A}-\widetilde{\mathcal{A}}\|_F$, where $M = \{\mathcal{C} = \mathcal{X}*\mathcal{Y} | \mathcal{X} \in \mathbb{R}^{n_1\times k\times n_3},\ \mathcal{Y} \in \mathbb{R}^{k\times n_2\times n_3}\}$.

Proof. We will use (3.1), unitary invariance of (partially) orthogonal tensors, and the definition of the T-SVD to complete the proof. Let $n = \min(n_1,n_2)$:

$$\begin{aligned}
\|\mathcal{A}-\mathcal{A}_k\|_F^2 &= \|\mathcal{S}(k+1{:}n, k+1{:}n, :)\|_F^2 \\
&= \|(F_{n_3}\otimes I)\mathrm{MatVec}(\mathcal{S}(k+1{:}n, k+1{:}n, :)\|_F^2 \\
&= n_3\|\Sigma_1(k+1{:}n, k+1{:}n)\|_F^2 + \cdots + n_3\|\Sigma_{n_3}(k+1{:}n, k+1{:}n)\|_F^2.
\end{aligned}$$

Now let $\mathcal{B} \in M$, so that $\mathcal{B} = \mathcal{X}*\mathcal{Y}^T$. Then

$$\begin{aligned}
\|\mathcal{A}-\mathcal{B}\|_F^2 &= \|\mathrm{MatVec}(\mathcal{A}) - \mathrm{circ}(\mathcal{X})\mathrm{MatVec}(\mathcal{Y}^T)\|_F^2 \\
&= \|(F_{n_3}\otimes I)\mathrm{MatVec}(\mathcal{A}) - (F_{n_3}\otimes I)\mathrm{circ}(\mathcal{X})(F_{n_3}^*\otimes I)(F_{n_3}\otimes I)\mathrm{MatVec}(\mathcal{Y}^T)\|_F^2 \\
&= n_3\|D_1 - \widehat{X}_1\widehat{Y}_1^T\|_F^2 + \cdots + n_3\|D_{n_3} - \widehat{X}_{n_3}\widehat{Y}_{n_3}^T\|_F^2 \\
&\geqslant n_3\|\Sigma_1(k+1{:}n, k+1{:}n)\|_F^2 + \cdots + n_3\|\Sigma_{n_3}(k+1{:}n, k+1{:}n)\|_F^2. \qquad \square
\end{aligned}$$

Thus, it appears the straightforward way to compress the tensor is to choose some $k < \min(n_1,n_2)$ and compute

$$\mathcal{A} \approx \sum_{i=1}^{k} \mathcal{U}(:,i,:)*\mathcal{S}(i,i,:)*\mathcal{V}(:,i,:)^T. \tag{4.6}$$

Unfortunately, it is not immediately obvious that (4.6) leads to a very compressed representation. At first glance, the method requires the storage of $\mathcal{U}(:,i,:)$ for $i = 1,\ldots,k$, so $k, n_1\times n_2$ matrices, and storage of $\mathcal{S}(i,i,:)V(:,i,:)^T$, so $k, n_2\times n_3$ matrices. Even if k is small, the memory storage is prohibitive.

The columns of the matrix $\mathcal{U}(:,i,:)$ may be nearly linearly dependent. To see this, observe that if $\mathcal{U}(:,i,:)*\mathcal{S}(i,i,:)*\mathcal{V}(:,i,:)^T$ is a rank-1 tensor, $\mathcal{S}(i,i,:)*V(:,i,:)^T$ and $\mathcal{U}(:,i,:)$ must each have rank 1.

Thus, if this term is well approximated by a tensor of low rank, we expect this to be reflected in singular values of each of the matrices $\mathcal{U}(:,i,:)$ and $\mathcal{S}(i,i,:)*\mathcal{V}(:,i,:)^T$.

Therefore, one practical compression strategy is to take (4.6) and for each i, compute a low rank approximation to $\mathcal{U}(:,i,:)*\mathcal{S}(i,i,:)*\mathcal{V}(:,i,:)^T$. There are several ways this could be computed. We consider one method here.

Consider that for each i we have

$$\mathcal{U}(:,i,:) = \sum_{j=1}^{n} p^{(j)} \circ \mu^{(j)} \circ q^{(j)}, \quad \mathcal{S}(i,i,:)*\mathcal{V}(:,i,:)^T = \sum_{j=1}^{n} \lambda^{(j)} \circ b^{(j)} \circ t^{(j)},$$

where $\mu^{(j)}$, $\lambda^{(j)}$ are scalars. These could be given by the matrix SVD's of appropriately oriented $\mathcal{U}(:,i,:)$, $\mathcal{S}(i,i,:)*\mathcal{V}(:,i,:)^T$, for example. Thus, their product is

$$\mathcal{U}(:,i,:)*\mathcal{S}(i,i,:)*\mathcal{V}(:,i,:)^T = \sum_{j=1}^{n}\sum_{\ell=1}^{n} \mu^{(j)}\lambda^{(\ell)}(p^{(j)} \circ b^{(\ell)} \circ \mathrm{circ}(q^{(j)})t^{(\ell)}). \tag{4.7}$$

This is an outer product representation of each tensor in the sum (4.6). From the right of (4.7) for each $i = 1,\ldots,k$, we wish to drop certain terms. Appealing to Lemma 4.2, denote $\sigma_i = \sum_{k=1}^{n_3} \mathcal{S}(i,i,k)$. Since σ_i are the singular values by the lemma, this suggests we drop any terms for which

$$\sigma_i \mu^{(j)} \lambda^{(\ell)} \|\mathrm{circ}(q^{(j)})t^{(\ell)}\|_\infty < tol.$$

The proposed algorithm for computing the resulting approximation to $\mathcal{A}$ is below. The approximation is returned in Kruskal form (i.e., $\mathcal{A} \approx [[U,V,W]]$). Note the parallelizability in that the full T-SVD need not be approximated at the start of the algorithm.

Algorithm T-Compress

Input: $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$, truncation index k
for $i = 1,\ldots,k$
(1) Compute $\mathcal{U}(:,i,:)$, $\mathcal{S}(i,i,:)$, $\mathcal{V}(:,i,:)$ if not already available
(2) Compute k_1 terms of the SVD for $\mathcal{U}(:,i,:)$ and k_2 terms of the SVD for $\mathcal{S}(i,i,:)*\mathcal{V}(:,i,:)$
(3) for $j = 1:k_1$
for $\ell = 1:k_2$
if $\sigma_i \mu^{(j)} \lambda^{(\ell)} \|\mathrm{circ}(q^{(j)})t^{(\ell)}\|_\infty > tol$,
$U = [U, p^{(j)}]$, $V = [V, b^{(\ell)}]$, $W = [W, \mu^{(j)}\lambda^{(\ell)}\mathrm{circ}(q^{(j)})t^{(\ell)}]$

T-Compress relies on computing at least k terms of the T-SVD, although this stage can be interleaved with computing the rest of the approximation. However, Theorem 4.3 suggests that computing the T-SVD is not necessary in practice. We can modify steps 1 and 2 to obtain the following.

Algorithm T-Compress, ver. 2

Input: $n_1 \times n_2 \times n_3$ tensor $\mathcal{A}$, truncation index k
Initialize $\mathcal{A}_{curr} = \mathcal{A}$.
for $i = 1,\ldots,k$
(1) Compute $\mathcal{G} \in \mathbb{R}^{n_1 \times 1 \times n_3}$, $\mathcal{H} \in \mathbb{R}^{1 \times n_2 \times n_3}$ as $\mathcal{G}$, $\mathcal{H} = \arg\min \|\mathcal{A} - \mathcal{G}*\mathcal{H}\|_F$
(2) Compute k_1 terms of the SVD for $\mathcal{G}$ and k_2 terms of the SVD for $\mathcal{H}$
(3) for $j = 1:k_1$
for $\ell = 1:k_2$
if $\sigma_i \mu^{(j)} \lambda^{(\ell)} \|\mathrm{circ}(q^{(j)})t^{(\ell)}\|_\infty > tol$,
$U = [U, p^{(j)}]$, $V = [V, b^{(\ell)}]$, $W = [W, \mu^{(j)}\lambda^{(\ell)}\mathrm{circ}(q^{(j)})t^{(\ell)}]$
(4) $\mathcal{A}_{curr} = \mathcal{A}_{curr} - \mathcal{G}*\mathcal{H}$

Of course, there are many different variations on the idea we have just presented (e.g. using an 'optimal' approximation in place of SVDs of the individual matrices, adding a step that checks if one

can "shrink" the rank of the approximation, taking $\mathcal{G}$, $\mathcal{H}$ to have more than 1 slice), but we do not wish to pursue them all here in the interest of space.

One should compare T-Compress to algorithms that seek to find low-rank approximations of tensors by subtracting off "best rank-1" approximations one after the other. It has been shown (see [42], for example) that subtracting a best rank-1 tensor approximation from $\mathcal{A}$ does not necessarily reduce the rank: that is, if $\mathcal{A}$ has rank r, and $u \circ v \circ w$ is the best rank-1 approximation to $\mathcal{A}$, $\mathcal{A} - u \circ v \circ w$ does not have to have rank $r - 1$. This is true for our algorithm as well: $\mathcal{A}_{curr}$ may not have lower rank. From (4.7), if $\|\mathcal{U}(:,i,:) * \mathcal{S}(i,i,:) * \mathcal{V}(:,i,:)^T\|_F \geqslant \|\sum_{j\notin J}\sum_{\ell\notin L}\mu^{(j)}\lambda^{(\ell)}(p^{(j)} \circ b^{(\ell)} \circ \mathrm{circ}(q^{(j)})t^{(\ell)})\|_F$, where J, L denote (for fixed i) the indicies of terms that were kept each sum on the right, the residual between $\mathcal{A}$ and the tensor approximation (output of the algorithm) must be non-increasing as a function of k. This seems to be born out in our examples, but more study is needed.

5. An application from image processing

In this section, we illustrate the potential utility of our new tensor–product formulation and related definitions on an application in image processing.

The discrete model for 2D image blurring is represented as

$$Ax = b,$$

where A is known as the blurring operator, x is the "image" unstacked by columns to obtain a vector, and b is a column vector representing the image. In truth, b has been corrupted by some noise, A is ill-conditioned, so even if A is theoretically invertible, the exact solution x will be contaminated by noise.

The regularization procedure used to generate an approximation to the desired image is iterative, and to speed convergence to this solution requires a preconditioner; that is, a matrix M such that AM has some of the singular values (corresponding to the so-called signal subspace) near 1, but which leaves the noise subspace (corresponding to the small singular values of A) untouched [18]. Then one applies the iterative method to the system

$$AMy = b, \quad x = M^{\dagger}y. \tag{5.1}$$

At each step of the iterative method, one will have to compute matrix–vector products with A and M (possibly also A^T, M^T, depending on which method is used), and therefore matrix–vector products with M need to be performed efficiently. Indeed, the cost of using an iterative regularization method is roughly the sum of the costs of these matrix–vector products times the number of iterations needed to reach the solution.

A preconditioner M with the desired SVD spectral clustering properties could be easily obtained from the SVD of A if it were available; the small singular values are replaced by 1, and M is obtained as the (psuedo)inverse of the result. Unfortunately, it is usually too costly to factor A to obtain the desired rank-revealing information needed to generate M directly, nor would matrix–vector products with M so defined be efficient. It is common in image deblurring applications to assume that the blurring matrix A has some structure: for example, it might be block Toeplitz with Toeplitz blocks (BTTB). In such cases, a reasonable first step is to compute a level-1 circulant approximation of A (i.e., a block circulant approximation), called $\widetilde{A}$. This can be block diagonalized by a 1D Fourier transform, and then one may work in Fourier space to define the preconditioner M from $\widetilde{A}$ [13,18].

Here, we want to exploit the fact that since the approximate blurring matrix $\widetilde{A}$ is block circulant, the corresponding approximate blurring model $\widetilde{A}x = b$ can be written in terms of a third order tensor (the approximate blurring operator) acting on a matrix (e.g. the image) through the use of the t-product:

$$\widetilde{\mathcal{A}} * \mathcal{X} = \mathcal{B},$$

where $\mathcal{X} = \mathrm{fold}(x)$, $\mathcal{B} = \mathrm{fold}(b)$. Thus, we can define our preconditioner from (an approximation to) the operator $\widetilde{\mathcal{A}}$ itself. The approximation is obtained from Algorithm T-compress in Section 4.1. We then generate a regularized pseudoinverse from the output, and this gives our preconditioner, except that the matrix M is not available explicitly. Rather, the resulting preconditioner will also be represented in terms of a tensor, $\mathcal{M}$, and so the matrix–vector product Mv is computed as $\mathcal{M} * \mathrm{fold}(v)$.

Here, we assume A is square, with n blocks of size $n \times n$, and that A is BTTB. The matrix is generated from the point-spread function (see, for example, [33] for how to generate such a matrix from the PSF), which in turn is the sum of three, nonsymmetric Gaussian blurring kernels with different variances. Code to generate our PSF is given below. We let $\widetilde{A}$ denote the T-Chan level-1 block circulant with Toeplitz blocks (BCTB) matrix approximation to A (see [8]).

In this example, we use $n = 128$. First, we create an approximation

$$\widetilde{\mathcal{A}} \approx \sum_{i=1}^{k} u^{(i)} \circ v^{(i)} \circ w^{(i)}, \tag{5.2}$$

following Algorithm T-Compress in Section 4.1 by setting $k = 50,\ k_1 = k_2 = 1$.

Our goal now is to use this output $[[U, V, W]]$ to produce another tensor, $\mathcal{M}$, which is a regularized inverse of $\widetilde{\mathcal{A}}$ in the sense that when applying it to $\mathcal{B}$, the resulting image should resemble a less blurred (without noise amplification) of the image. Once such $\mathcal{M}$ has been identified, we will iterate on the right preconditioned system (5.1), again noting that we compute the multiplication of M with a vector v as $\mathcal{M}*\text{fold}(v)$. It can be shown that if the Kruskal form of $\mathcal{M}$ is available, this product can be performed in only $O(kn^2 + kn\lg(n))$ flops. This means an application of the preconditioner per iteration is on the order of the $O(n^2\lg(n))$ cost of a matrix–vector product[4] with A. So if the number of iterations to achieve the regularized solution of the preconditioned system is significantly smaller than it is to compute the regularized solution to the unpreconditioned system, we have an efficient deblurring algorithm.

To generate $\mathcal{M}$, we will use Definition 3.12 with one small adjustment. In our case, the condition numbers of U and V are near 1. However, W has many small magnitude (numerically zero) Fourier coefficients. Because of this, Definition 3.12 cannot directly be applied. Instead, $\mathcal{M} = [[X, Y, Z]]$ where X, Y are determined as in 3.12 and Z is determined as follows. For the Fourier coefficients

$$\hat{z}_j^{(i)} = \begin{cases} 1/\hat{w}_j^{(i)} & \hat{w}_j^{(i)} < \gamma_c, \\ 1 & \hat{w}_j^{(i)} \geqslant \gamma_c. \end{cases}$$

We chose our threshold γ_c by trial and error visually by inspecting $\mathcal{M}*\mathcal{B}$.

The PSF was created using the following MATLAB script:

```
T = gausswin(m, 20); T2 = gausswin(m, 25); Psf1 = reshape(kron(T, T2), m, m);
T = gausswin(m, 27); T2 = gausswin(m, 23); Psf2 = reshape(kron(T, T2), m, m);
T = gausswin(m,23); T2 = gausswin(m, 30); Psf3 = reshape(kron(T, T2), m, m);
PSF = Psf1 + Psf2 + Psf3.
```

For this example, the true image is a 128×128 downsampled (scaled) version of the satellite image, in the left of Fig. 5.1. We then formed $b = Ax$, where x is the vectorized version of the true image. Gaussian white noise was added to b so that the noise level was 0.1%. The blurred, noisy image is in the middle of Fig. 5.1. As mentioned above, we viewed $\mathcal{M}*\mathcal{B}$ where $\mathcal{M}$ is defined for three different choices of γ_c, 0.1, 0.5, 1. We chose $\gamma_c = 0.5$ since the other two values seemed to give an image that was underregularized or overregularized, respectively. A more sophisticated mechanism for choosing the regularization parameter γ_c would be necessary in practice; the reader is referred to [14] for one possibility. The reconstruction obtained using 3 iterations of the LSQR[5] algorithm [36] applied to the preconditioned problem is shown on the right of Fig. 5.1. Three iterations corresponded to the optimal (in the two-norm) reconstruction as compared to the true image. This was obtained via our MATLAB codes in 0.126 s. A solution of comparable quality, as measured in the 2-norm of the error with the exact solution, was obtained with unpreconditioned LSQR in 325 iterations and required 8.52 s to compute.

We believe this example illustrates the potential of many new ideas presented in this paper (t-product, T-SVD, pseudoinverses, and our compression strategy) in at least one application in image

[4] Products with A are computed by embedding in a BCCB matrix and using 2D FFTs.

[5] LSQR is a Krylov-subspace iterative method for solving the least squares problem. It is known to act as a regularization method if iterations are stopped before the least squares solution is reached.

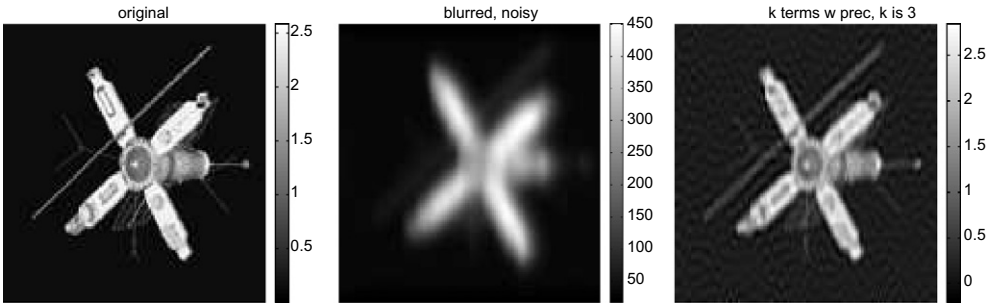

Fig. 5.1. True image (left), blurred noisy image (middle), reconstruction after 3 iterations of preconditioned LSQR with $\mathcal{M}$ defined using $\gamma_c = 0.5$.

processing. Our current work suggests that our approach might also be valuable in the context of facial recognition.

For different strategies and models relating tensors to deblurring see [32,37].

6. Conclusions and future work

In order to determine compressed representations of tensors, we introduced the notion of a t-product between tensors. We subsequently derived formulations of tensor identity, inverse, pseudoinverse, and transpose. We showed that the set of $n \times n \times n$ tensors with the t-product, inverse and identity forms a group. We also showed that the t-product defines a linear operator, and discussed its range and null space. Furthermore, we showed that using the t-product we could extend such orthogonal matrix factorizations such as the SVD and QR factorizations to tensors. The resulting T-SVD gave a means for optimally approximating the tensor as a sum of outer products of matrices. We then proposed an approximation algorithm for the tensor based on this k-term optimal sum. We demonstrated the utility of our approximation algorithm, as well as the utility of concepts such as right pseudoinverse, on an application from image deblurring.

Our focus in this paper was on developing a representation specifically for third-order tensors. However, our approach naturally generalizes to higher-order tensors in a recursive manner. The interpretation of range discussed in this paper leads us to consider extensions of the concept of Krylov iterative methods. In future work, we will explore the possibility of devising non-negative tensor factorizations based on our t-product approach. Our results regarding the null space suggest it might be possible to look for sparse (e.g. compressed) approximations by adding constraints to the Fourier domain.

Acknowledgements

We wish to thank the anonymous reviewers whose comments resulted in a significantly improved manuscript.

References

[1] E. Acar, S. Çamtepe, M. Krishnamoorthy, B. Yener, Modeling and multiway analysis of chatroom tensors, in: Proceedings of the IEEE International Conference on Intelligence and Security Informatics, EMBS 2007, Lecture Notes in Computer Science, vol. 3495, 2005, pp. 256–268.
[2] B. Bader, T. Kolda, Algorithm 862: MATLAB tensor classes for fast algorithm prototyping, ACM Trans. Math. Software 32 (2006) 635–653.
[3] C. Beckmann, S. Smith, Tensorial extensions of the independent component analysis for multisubject FMRI analysis, NeuroImage 25 (2005) 294–311.
[4] J.T. Berge, Kruskal's polynomial for $2 \times 2 \times 2$ arrays and a generalization to $2 \times n \times n$ arrays, Psychometrika 56 (1991) 631–636.
[5] K. Braman, Third-order tensors as linear operators on a space of matrices, Linear Algebra Appl. 433 (2010) 1241–1253.

[6] R. Bro, S.D. Jong, A fast non-negativity-constrained least squares algorithm, J. Chemom. 11 (1997) 393–401.
[7] J. Carroll, J. Chang, Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition, Psychometrika 35 (1970) 283–319.
[8] R. Chan, M. Ng, Conjugate gradient methods for Toeplitz systems, SIAM Rev. 38 (1996) 427–482.
[9] P. Comon, Tensor decompositions, in: J.G. McWhirter, I.K. Proudler (Eds.), Mathematics in Signal Processing V, Clarendon Press, Oxford, UK, 2002, pp. 1–24.
[10] J. Denis, T. Dhorne, Orthogonal tensor decomposition of 3-way tables, in: R. Coppi, S. Bolasco (Eds.), Multiway Data Analysis, Elsevier, Amsterdam, 1989, pp. 31–37.
[11] M.P. Friedlander, K. Hatz, Computing nonnegative tensor factorizations, Tech. Report TR-2006-21, University of British Columbia, Computer Science Department, 2006.
[12] G. Golub, C.V. Loan, Matrix Computations, third ed., Johns Hopkins University Press, Baltimore, 1996.
[13] M. Hanke, J.G. Nagy, R.J. Plemmons, Preconditioned iterative regularization, in: Numerical Linear Algebra, de Gruyter, Berlin, 1993, pp. 141–163.
[14] P.C. Hansen, M.E. Kilmer, R. Kjeldsen, Exploiting residual information in the parameter choice for discrete ill-posed problems, BIT 46 (2006) 41–59.
[15] R. Harshman, Foundations of the parafac procedure: model and conditions for an 'explanatory' multi-mode factor analysis, UCLA Working Papers in Phonetics, 16, 1970, pp. 1–84.
[16] J.J. Ja', Optimal evaluation of pairs of bilinear forms, SIAM J. Comput. 8 (1979) 443–461.
[17] H.A. Kiers, Towards a standardized notation and terminology in multiway analysis, J. Chemom. 14 (2000) 105–122.
[18] M.E. Kilmer, Cauchy-like preconditioners for 2-dimensional ill-posed problems, SIAM J. Matrix Anal. Appl. 20 (1999)
[19] M.E. Kilmer, C.D. Martin, L. Perrone, A third-order generalization of the matrix svd as a product of third-order tensors, Tech. Report TR-2008-4, Tufts University, Computer Science Department, 2008.
[20] E. Kofidis, P. Regalia, On the best rank-1 approximation of higher-order supersymmetric tensors, SIAM J. Matrix Anal. Appl. 23 (2001) 863–884.
[21] T. Kolda, B. Bader, Higher-order web link analysis using multilinear algebra, in: Proceedings of the Fifth IEEE International Conference on Data Mining, ICDM 2005, IEEE Computer Society, 2005, pp. 242–249.
[22] T. Kolda, B. Bader, Tensor decompositions and applications, SIAM Rev. 51 (2009) 455–500.
[23] T.G. Kolda, Multilinear operators for higher-order decompositions, Tech. Report SAND2006-2081, Sandia National Laboratories, 2006.
[24] P. Kroonenberg, Three-mode Principal Component Analysis: Theory and Applications, DSWO Press, Leiden, 1983.
[25] J. Kruskal, Rank, decomposition, and uniqueness for 3-way and n-way arrays, in: R. Coppi, S. Bolasco (Eds.), Multiway Data Analysis, Elsevier, Amsterdam, 1989, pp. 7–18.
[26] L.D. Lathauwer, B.D. Moor, J. Vandewalle, A multilinear singular value decomposition, SIAM J. Matrix Anal. Appl. 21 (2000) 1253–1278.
[27] L.D. Lathauwer, B.D. Moor, From matrix to tensor: multilinear algebra and signal processing, in: J. McWhirter, e.I. Proudler (Eds.), Mathematics in Signal Processing IV, Clarendon Press, Oxford, UK, 1998, pp. 1–15.
[28] L.D. Lathauwer, B.D. Moor, J. Vandewalle, On the best rank-1 and rank-$R_1, \ldots, R_N$ approximation of higher-order tensors, SIAM J. Matrix Anal. Appl. 21 (2000) 1324–1342.
[29] C. Martin, The rank of a $2 \times 2 \times 2$ tensor, Linear and Multilinear Algebra, submitted for publication.
[30] E. Martínez-Montes, P. Valdés-Sosa, F. Miwakeichi, R. Goldman, M. Cohen, Concurrent EEG/FMRI analysis by multiway partial least squares, NeuroImage 22 (2004) 1023–1034.
[31] F. Miwakeichi, E. Martínez-Montes, P. Valdés-Sosa, N. Nishiyama, H. Mizuhara, Y. Yamaguchi, Decomposing EEG data into space–time–frequency components using parallel factor analysis, NeuroImage 22 (2004) 1035–1045.
[32] J. Nagy, M. Kilmer, Kronecker product approximation for preconditioning in three-dimensional imaging applications, IEEE Trans. Image Process. 15 (2006) 604–613.
[33] J.G. Nagy, K. Palmer, L. Perrone, Iterative methods for image deblurring: a matlab object oriented approach, Numer. Algorithms 36 (2004) 73–93.
[34] I.V. Oseledets, D.V. Savostianov, E.E. Tyrtyshnikov, Tucker dimensionality reduction of three-dimensional arrays in linear time, SIAM J. Matrix Anal. Appl. 30 (2008) 939–956.
[35] P. Paatero, Construction and analysis of degenerate parafac models, J. Chemom. 14 (2000) 285–299.
[36] C.C. Paige, M.A. Saunders, LSQR: An algorithm for sparse linear equations and sparse least squares, ACM Trans. Math. Software 8 (1982) 43–71.
[37] M. Rezghi, L. Eldèn, Diagonalization of tensors with circulant structure, Linear Algebra Appl. 435 (2011) 422–447.
[38] B. Savas, L. Eldén, Handwritten digit classification using higher order singular value decomposition, Pattern Recognit. 40 (2007) 993–1003.
[39] N. Sidiropoulos, R. Bro, G. Giannakis, Parallel factor analysis in sensor array processing, IEEE Trans. Signal Process. 48 (2000) 2377–2388.
[40] V.D. Silva, L.-H. Kim, Tensor rank and the ill-posedness of the best low-rank approximation problem, SIAM J. Matrix Anal. Appl. 30 (2008) 1084–1127.
[41] A. Smilde, R. Bro, P. Geladi, Multi-way Analysis: Applications in the Chemical Sciences, Wiley, 2004.
[42] A. Stegeman, P. Comon, Subtracting a best rank-1 approximation may increase tensor rank, arXiv e-prints (2009).
[43] L. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika 31 (1966) 279–311.
[44] M. Vasilescu, D. Terzopoulos, Multilinear analysis of image ensembles: tensorfaces, in: Proceedings of the Seventh European Conference on Computer Vision, ECCV 2002, Lecture Notes in Computer Science, vol. 2350, 2002, pp. 447–460.
[45] M. Vasilescu, D. Terzopoulos, Multilinear image analysis for face recognition, in: Proceedings of the International Conference on Pattern Recognition, ICPR 2002, vol. 2, Quebec City, Canada, 2002, pp. 511–514.
[46] M. Vasilescu, D. Terzopoulos, Multilinear subspace analysis of image ensembles, in: Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2003, 2003, pp. 93–99.
[47] T. Zhang, G. Golub, Rank-one approximation to high order tensors, SIAM J. Matrix Anal. Appl. 23 (2001) 534–550.