SIAM J. MATRIX ANAL. APPL.
Vol. 34, No. 1, pp. 148–172

THIRD-ORDER TENSORS AS OPERATORS ON MATRICES: A THEORETICAL AND COMPUTATIONAL FRAMEWORK WITH APPLICATIONS IN IMAGING*

MISHA E. KILMER†, KAREN BRAMAN‡, NING HAO†, AND RANDY C. HOOVER§

Abstract. Recent work by Kilmer and Martin [*Linear Algebra Appl.*, 435 (2011), pp. 641–658] and Braman [*Linear Algebra Appl.*, 433 (2010), pp. 1241–1253] provides a setting in which the familiar tools of linear algebra can be extended to better understand third-order tensors. Continuing along this vein, this paper investigates further implications including (1) a bilinear operator on the matrices which is nearly an inner product and which leads to definitions for length of matrices, angle between two matrices, and orthogonality of matrices, and (2) the use of t-linear combinations to characterize the range and kernel of a mapping defined by a third-order tensor and the t-product and the quantification of the dimensions of those sets. These theoretical results lead to the study of orthogonal projections as well as an effective Gram–Schmidt process for producing an orthogonal basis of matrices. The theoretical framework also leads us to consider the notion of tensor polynomials and their relation to tensor eigentuples defined in the recent article by Braman. Implications for extending basic algorithms such as the power method, QR iteration, and Krylov subspace methods are discussed. We conclude with two examples in image processing: using the orthogonal elements generated via a Golub–Kahan iterative bidiagonalization scheme for object recognition and solving a regularized image deblurring problem.

Key words. eigendecomposition, tensor decomposition, singular value decomposition, multidimensional arrays, Krylov methods, tensor SVD

AMS subject classifications. 15A69, 65F30

DOI. 10.1137/110837711

1. Introduction.

The term *tensor*, as used in the context of this paper, refers to a multidimensional array of numbers, sometimes called an *n-way* or *n-mode* array. If, for example, $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, then we say $\mathcal{A}$ is a third-order tensor where *order* is the number of ways or modes of the tensor. Thus, matrices and vectors are second-order and first-order tensors, respectively. Third-order (and higher) tensors arise in a wide variety of application areas, including, but not limited to, chemometrics [30], psychometrics [20], and image and signal processing [6, 7, 29, 23, 13, 26, 34, 33, 35]. Various tensor decompositions such as CANDECOMP/PARAFAC (CP) [4, 12], TUCKER [32], and higher-order SVD [8] have been developed to facilitate the extension of linear algebra tools to this multilinear context. For a thorough review of tensor decompositions and their applications, see [18].

Recent work by Kilmer and Martin [16] and Braman [1] provides an alternative setting in which the familiar tools of linear algebra can be extended to better understand third-order tensors. Specifically, in [17] and [16] the authors define a multiplication operation which is closed on the set of third-order tensors. This multiplication allows tensor factorizations which are analogues of matrix factorizations

*Received by the editors June 17, 2011; accepted for publication (in revised form) October 26, 2012; published electronically February 28, 2013.
http://www.siam.org/journals/simax/34-1/83771.html
†Department of Mathematics, Tufts University, Medford, MA 02155 (misha.kilmer@tufts.edu, ning.hao@tufts.edu). The work of these authors was supported by NSF grant 0914957.
‡Department of Mathematics and Computer Science, South Dakota School of Mines and Technology, Rapid City, SD 57701 (karen.braman@sdsmt.edu).
§Department of Electrical and Computer Engineering, South Dakota School of Mines and Technology, Rapid City, SD 57701 (randy.hoover@sdsmt.edu).

such as SVD, QR, and eigendecompostions. In addition, [1] defines a free module of matrices (or $n \times 1 \times n$ tensors) over a commutative ring where the elements are vectors (or $1 \times 1 \times n$ tensors) and shows that *every* linear transformation upon that space can be represented by multiplication by a third-order tensor. Thus, the significant contribution of those papers is the development of a framework which allows new extensions of familiar matrix analysis to the multilinear setting while avoiding the loss of information inherent in *matricization* or *flattening* of the tensor.

Continuing along this vein, this paper investigates further implications of this new point of view. In particular, we develop new constructs that lead us ultimately (see section 6) to the extension of traditional Krylov methods to applications involving third order tensors. We are not the first to consider extensions of Krylov methods to tensor computation. Other recent work in this area includes the work in [27, 28]. Our methods are different from their work, however, because we rely on the t-product construct in [16] to generate the space in which we are looking for solutions, which makes more sense in the context of the applications we will discuss. The algorithms we present here are in the spirit of a proof-of-concept that is consistent with our new theoretical framework. The numerical results are intended to show the potential suitability in a practical sense. However, in order to maintain the paper's focus, the important but subtle and complicated numerical analysis of the algorithms in finite precision are left for future work.

The work in this paper includes results from the authors' technical report [15]. We should also note that there is overlap with the recent report [9], in which the authors adopt some of the same notational conventions for the spaces in which we work and in which the authors focus on computation of eigentuples as defined in [1]. We will make it clear in the text where there is consistency between the two bodies of work. The authors of [9] also have a freely available MATLAB toolbox in which some of what we propose in the present manuscript could readily be implemented.

This paper is organized as follows. After establishing basic definitions and notation in section 2, section 3 presents a bilinear operator on the module of matrices. This operator leads to definitions for length of matrices (which we prove in Appendix A also provides a new valid matrix norm), for a tube of angles between two matrices, and for orthogonality of matrices. In section 4 we introduce the concept of t-linear combinations in order to characterize the range and kernel of a third-order tensor and to quantify the dimensions of those sets, thus defining a tensor's *multirank* and *multinullity*. These theoretical results lead, in section 5, to the study of orthogonal projections and an effective Gram–Schmidt process for producing an orthogonal basis of matrices. In section 6 we consider powers of tensors, preliminary notes for computation of tensor eigendecompositions, and Krylov methods. In section 7, we utilize the constructs from the previous section in the context of two image processing problems: object recognition and image deblurring. We give concluding remarks and future work in section 8.

On a first pass, the reader may wish to bypass section 4 and proceed directly to the algorithms and numerical examples in sections 5–7. The details and definitions in section 4 provide the keys to understanding the performance and pitfalls of the algorithms and also complete the theoretical framework, but this material is not crucial in understanding the premise of the algorithms on a first read.

2. Preliminaries. In this section, we give the basic definitions from [16] and introduce the notation used in the rest of the paper. We will use calligraphic script

to denote third-order tensors, capital letters to denote matrices, and lowercase math font to denote scalars.

2.1. Notation and indexing. It will be convenient to break a tensor $\mathcal{A}$ in $\mathbb{R}^{\ell\times m\times n}$ up into various slices and tubal elements, and to have an indexing on those. The ith lateral slice will be denoted $\vec{\mathcal{A}}_i$, whereas the jth frontal slice will be denoted $A^{(j)}$. In terms of MATLAB indexing notation, this means $\vec{\mathcal{A}}_i \equiv \mathcal{A}(:,i,:)$, which is a tensor, while $A^{(j)} \equiv \mathcal{A}(:,:,j)$, which is a matrix.

We use the notation $\mathbf{a}_{ik}$ to denote the i,kth tube fiber in $\mathcal{A}$, that is, $\mathbf{a}_{ik} = \mathcal{A}(i,k,:)$. The jth entry in that tube is $\mathbf{a}_{ik}^{(j)}$. Indeed, these tubes have special meaning for us in the present work, as they will play a role similar to scalars in $\mathbb{R}$. Thus, we make the following definition.

DEFINITION 2.1. *An element $\mathbf{c} \in \mathbb{R}^{1\times 1\times n}$ is called a* tubal scalar *of length n. Note that $\mathbf{c}^{(j)}$ refers to a scalar, namely, the jth frontal face of the tubal scalar.*

Following this, we get a series of additional notation.

DEFINITION 2.2. *The space of all tubal scalars is denoted $\mathbb{K}_n$ (see also* [1, 9]*).*

DEFINITION 2.3. *Let $\mathcal{C} \in \mathbb{R}^{\ell\times 1\times n}$. Then $\mathcal{C}$ is a length ℓ vector of tubal scalars and will be denoted $\mathbb{K}_n^{\ell}$. Thus, we will use the notation $\vec{\mathcal{C}}$ to denote elements of $\mathbb{K}_n^{\ell}$.*

DEFINITION 2.4. *Let $\mathcal{C} \in \mathbb{R}^{\ell\times m\times n}$. Then $\mathcal{C}$ is an $\ell \times m$ matrix of tubal scalars and will be denoted by $\mathbb{K}_n^{\ell\times m}$.*

In order to discuss multiplication between two tensors and to understand the basics of the algorithms we consider here, we first must introduce the concept of converting $\mathcal{A} \in \mathbb{R}^{\ell\times m\times n}$ into a block circulant matrix.

If $\mathcal{A} \in \mathbb{R}^{\ell\times m\times n}$ with $\ell \times m$ frontal slices denoted $A^{(i)}$, then

$$\texttt{bcirc}(\mathcal{A}) = \begin{bmatrix} A^{(1)} & A^{(n)} & A^{(n-1)} & \dots & A^{(2)} \\ A^{(2)} & A^{(1)} & A^{(n)} & \dots & A^{(3)} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ A^{(n)} & A^{(n-1)} & \ddots & A^{(2)} & A^{(1)} \end{bmatrix}$$

is a block circulant matrix of size $\ell n \times mn$.

We anchor the `unfold` command to the frontal slices of the tensor. That is, $\texttt{unfold}(\mathcal{A})$ takes an $\ell \times m \times n$ tensor and returns a block $\ell n \times m$ matrix, whereas the `fold` command undoes the operation:

$$\texttt{unfold}(\mathcal{A}) = \begin{bmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(n)} \end{bmatrix}, \qquad \texttt{fold}(\texttt{unfold}(\mathcal{A})) = \mathcal{A}.$$

When we are dealing with matrices, the `vec` command unwraps the matrix into a vector by column stacking, so that in MATLAB notation $\texttt{vec}(A) \equiv A(:)$.

2.2. t-Product, identity, inverse. The following definition of the t-product between two tensors was introduced in [17, 16].

DEFINITION 2.5. *Let $\mathcal{A}$ be $\ell \times p \times n$ and $\mathcal{B}$ be $p \times m \times n$. Then the t-product $\mathcal{A} * \mathcal{B}$ is the $\ell \times m \times n$ tensor*

$$\mathcal{A} * \mathcal{B} = \texttt{fold}\left(\texttt{bcirc}(\mathcal{A}) \cdot \texttt{unfold}(\mathcal{B})\right).$$

Note that in general the t-product of two tensors will not commute. There is one special exception in which the t-product always commutes: the case when $\ell = p = m = 1$, that is, when the tensors are tubal scalars.

We give an example of the t-product that will help illustrate a fundamental fact going forward, namely, that *third-order tensors under the t-product can be considered as operators on matrices* when those matrices are twisted into the third dimension and considered themselves as third-order tensors.

Example 2.6. Let $\mathcal{A} \in \mathbb{K}_2^{3\times 2}$ with frontal faces

$$A^{(1)} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ -1 & 3 \end{bmatrix} \qquad \text{and} \qquad A^{(2)} = \begin{bmatrix} -2 & 1 \\ -2 & 7 \\ 0 & -1 \end{bmatrix},$$

and let $\vec{\mathcal{B}} \in \mathbb{K}_2^2$ with frontal faces $B^{(1)} = \left[\begin{smallmatrix} 3 \\ -1 \end{smallmatrix}\right]$ and $B^{(2)} = \left[\begin{smallmatrix} -2 \\ -3 \end{smallmatrix}\right]$. Note that $\vec{\mathcal{B}}$ *is a* 2×2 *matrix oriented as the sole lateral slice of a third-order tensor.* Then

$$\mathcal{A} * \vec{\mathcal{B}} = \texttt{fold}\left(\left[\begin{array}{cc|cc} 1 & 0 & -2 & 1 \\ 0 & 2 & -2 & 7 \\ -1 & 3 & 0 & -1 \\ \hline -2 & 1 & 1 & 0 \\ -2 & 7 & 0 & 2 \\ 0 & -1 & -1 & 3 \end{array}\right]\left[\begin{array}{c} 3 \\ -1 \\ \hline -2 \\ -3 \end{array}\right]\right) = \texttt{fold}\left(\left[\begin{array}{c} 4 \\ -19 \\ -3 \\ \hline -9 \\ -19 \\ -6 \end{array}\right]\right) \in \mathbb{K}_2^{3\times 2}$$

is a $3 \times 1 \times 2$ tensor. In other words, in this example, $\vec{\mathcal{C}} := \mathcal{A} * \vec{\mathcal{B}}$ is a 3×2 matrix, oriented as a lateral slice of a third-order tensor.

Before moving on, we need a few more definitions from [16] and examples.

DEFINITION 2.7. *If $\mathcal{A}$ is $\ell \times m \times n$, then $\mathcal{A}^T$ is the $m \times \ell \times n$ tensor obtained by transposing each of the frontal slices and then reversing the order of transposed frontal slices* 2 *through* n.

DEFINITION 2.8. *The $m \times m \times n$ identity tensor $\mathcal{I}_{mmn}$ is the tensor whose first frontal slice is the $m \times m$ identity matrix, and whose other frontal slices are all zeros.*

As a special case, the $1 \times 1 \times n$ identity tensor is the tubal scalar $\mathbf{e}_1$ with a 1 in the first frontal face, and zeros in the remaining faces. For $m > 1$, $\mathcal{I}_{mmn}$ is a diagonal matrix of tubal scalars with diagonal elements all equal to $\mathbf{e}_1$.

DEFINITION 2.9. *An $m \times m \times n$ real-valued tensor $\mathcal{Q}$ is orthogonal if $\mathcal{Q}^T * \mathcal{Q} = \mathcal{Q} * \mathcal{Q}^T = \mathcal{I}$.*

For an $m \times m \times n$ tensor, an inverse exists if it satisfies the following.

DEFINITION 2.10. *An $m \times m \times n$ tensor $\mathcal{A}$ has an inverse $\mathcal{B}$, provided that*

$$\mathcal{A} * \mathcal{B} = \mathcal{I}_{mmn}, \qquad \text{and} \qquad \mathcal{B} * \mathcal{A} = \mathcal{I}_{mmn}.$$

(Note that, due to the definition of the $*$ operation, invertibility of $\mathcal{A}$ is equivalent to invertibility of $\texttt{bcirc}(\mathcal{A})$.) When it is not invertible, then, as we will see, the kernel of the operator induced by the t-product will be nontrivial.

2.3. A new view on tensors. One observation that is central to the work presented here is that tensors in $\mathbb{R}^{m\times 1\times n}$ (i.e., elements of $\mathbb{K}_n^m$) are simply matrices in $\mathbb{R}^{m\times n}$, oriented laterally. (See Figure 2.1.)

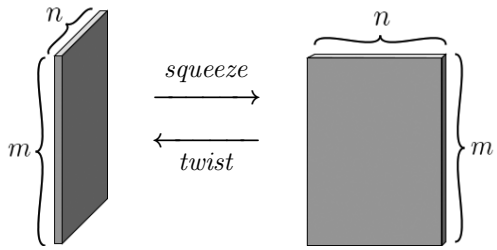

FIG. 2.1. $m \times 1 \times n$ *tensors and* $m \times n$ *matrices related through the* squeeze *and* twist *operations.*

To go back and forth between elements in $\mathbb{K}_n^m$ and elements of $\mathbb{R}^{m \times n}$, we introduce two operators: squeeze and twist.[1]

$$X = \texttt{squeeze}(\vec{\mathcal{X}}) \Rightarrow X(i,j) = \vec{\mathcal{X}}(i,1,j); \qquad \texttt{twist}(\texttt{squeeze}(\vec{\mathcal{X}})) = \vec{\mathcal{X}}.$$

Let $\vec{\mathcal{X}} := \texttt{twist}(X) \in \mathbb{K}_n^m$, $\mathbf{c} \in \mathbb{K}_n$. Then it is straightforward, using the definition of t-product, to show the following equivalence:

$$\texttt{squeeze}(\vec{\mathcal{X}} * \mathbf{c}) = X\texttt{bcirc}(\mathbf{c}^T). \tag{2.1}$$

2.4. Mapping matrices to matrices. In the context of the matrix factorizations presented in [16] (see also [1]), the authors also noted the following when $p > 1$.

OBSERVATION 2.11. $\mathcal{A} * \mathcal{B} = [\mathcal{A} * \vec{\mathcal{B}}_1, \mathcal{A} * \vec{\mathcal{B}}_2, \ldots, \mathcal{A} * \vec{\mathcal{B}}_p]$.

Thus it is natural to specifically consider the action of third-order tensors on matrices, where those matrices are oriented as $m \times 1 \times n$ tensors. In particular, this means that the t-product

$$T(\vec{\mathcal{X}}) = \mathcal{A} * \vec{\mathcal{X}}$$

defines a linear operator (in the traditional sense) from the set of all $m \times n$ matrices to $\ell \times n$ matrices, where those matrices are oriented laterally. Moreover, T describes a t-linear operator from $\mathbb{K}_n^m \to \mathbb{K}_n^\ell$ when elements of $\mathbb{K}_n$ are used in place of traditional scalars (that is, $T(\vec{\mathcal{X}} * \mathbf{c} + \vec{\mathcal{Y}} * \mathbf{d}) = T(\vec{\mathcal{X}}) * \mathbf{c} + T(\vec{\mathcal{Y}}) * \mathbf{d}$ for arbitrary tubal scalars $\mathbf{c}, \mathbf{d}$ and arbitrary $\vec{\mathcal{X}}, \vec{\mathcal{Y}} \in \mathbb{K}_n^m$). If $\ell = m$, then T will be invertible when $\mathcal{A}$ is invertible.

2.5. The Fourier domain connection. From a theoretical and a practical point of view, it now behooves us to understand the role of the Fourier transform in this setting. It is well known that block circulant matrices can be block diagonalized by using the Fourier transform. Mathematically, this means that if F_n denotes the $n \times n$ (normalized) DFT matrix, then for $\mathcal{A} \in \mathbb{R}^{\ell \times m \times n}$, there exist n, $\ell \times m$ matrices $\hat{A}^{(i)}$, possibly with complex entries, such that

$$(F_n \otimes I_\ell)\texttt{bcirc}(\mathcal{A})(F_n^H \otimes I_m) = \texttt{blockdiag}(\hat{A}^{(1)}, \ldots, \hat{A}^{(n)}) = \begin{bmatrix} \hat{A}^{(1)} & 0 & \cdots & 0 \\ 0 & \hat{A}^{(2)} & 0 & \cdots \\ \vdots & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & \hat{A}^{(n)} \end{bmatrix}. \tag{2.2}$$

[1] The squeeze operation works on a tensor $\vec{\mathcal{X}}$ in $\mathbb{R}^{m \times 1 \times n}$ just as it does in MATLAB. Our thanks to Tamara Kolda for suggesting the twist operator.

Fortunately, it is *not necessary* to form $\texttt{bcirc}(\mathcal{A})$ explicitly to generate the matrices $\hat{A}^{(i)}$. Using MATLAB notation, define $\hat{\mathcal{A}} := \texttt{fft}(\mathcal{A}, [\,], 3)$ as the tensor obtained by applying the FFT along each tubal element of $\mathcal{A}$. Then $\hat{A}^{(i)} := \hat{A}(:, :, i)$. For the remainder of the paper, hat notation is used to indicate that we are referencing the object after having taken an FFT along the third dimension.

The consequence of (2.2) is that t-products, as well as tensor factorizations, can be computed in the Fourier domain. For more details, see [17] and [16]. We cover two important examples here.

OBSERVATION 2.12. *Given* $\mathbf{a}, \mathbf{b} \in \mathbb{K}_n$, $\mathbf{a} * \mathbf{b}$ *can be computed as*

$$\mathbf{a} * \mathbf{b} := \texttt{ifft}(\hat{\mathbf{a}} \odot \hat{\mathbf{b}}, [\,], 3),$$

where $\odot$ *of two tubal scalars means pointwise multiplication.*

OBSERVATION 2.13. *Factorizations of* $\mathcal{A}$*, such as the t-QR and t-SVD (written* $\mathcal{A} = \mathcal{Q} * \mathcal{R}$ *and* $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$*, respectively) are created (implicitly) by applying the appropriate matrix factorization to each of the* $\hat{\mathcal{A}}^{(i)}$ *on the block diagonal of the right-hand side of* (2.2)*. For example,*

$$\mathcal{A} = \mathcal{Q} * \mathcal{R} \Longleftrightarrow \hat{\mathcal{A}}^{(i)} = \hat{\mathcal{Q}}^{(i)} \hat{\mathcal{R}}^{(i)}.$$

The MATLAB-like pseudocode for computing this t-QR factorization is given in Algorithm 1*. Other third-order tensor factorizations based on the t-product can be computed similarly. It is possible to show that* $\mathcal{Q}$ *arising from the t-QR factorization gives an example of an orthogonal tensor.*

Algorithm 1: Full tensor QR factorization, Fourier domain computation.

Input: $\mathcal{A} \in \mathbb{K}_n^{\ell \times p}$, $\ell \geq p$
Output: $\mathcal{Q} \in \mathbb{K}_n^{\ell \times p}, \mathcal{R} \in \mathbb{K}_n^{p \times p}$ such that $\mathcal{A} = \mathcal{Q} * \mathcal{R}$

$\hat{\mathcal{A}} \leftarrow \text{fft}(\mathcal{A}, [\,], 3)$;
for $i = 1$ **to** n
  Factor $\hat{\mathcal{A}}(:, :, i) = QR$ where Q is unitary;
  $\hat{\mathcal{Q}}(:, :, i) \leftarrow Q; \hat{\mathcal{R}}(:, :, i) \leftarrow R$
end for
$\mathcal{Q} \leftarrow \text{ifft}(\hat{\mathcal{Q}}, [\,], 3); \mathcal{R} \leftarrow \text{ifft}(\hat{\mathcal{R}}, [\,], 3)$;

For reference, we recall the conjugate symmetry of the Fourier transform. That is, if $\mathbf{v} \in \mathbb{K}_n$, then $\hat{\mathbf{v}}$ satisfies the following:

1. If n is even, then for $i = 2, \ldots, n/2$, $\hat{\mathbf{v}}^{(i)} = \texttt{conj}(\hat{\mathbf{v}}^{(n-i+2)})$.
2. If n is odd, then for $i = 2, \ldots, (n+1)/2$, $\hat{\mathbf{v}}^{(i)} = \texttt{conj}(\hat{\mathbf{v}}^{(n-i+2)})$.

This means for, say, n odd and $i > 1$ that $\hat{\mathcal{A}}^{(i)} = \texttt{conj}(\hat{\mathcal{A}}^{(n-i+2)})$. This provides a savings in computational time in computing tensor products in the Fourier domain or computing a tensor factorization in the tensor domain: for example, to compute $\mathcal{A} = \mathcal{Q} * \mathcal{R}$, we would only need to compute individual matrix QRs for about half the faces of $\hat{\mathcal{A}}$.

It is particularly interesting and illustrative to consider the t-product $\mathcal{C} = \mathcal{A}^T * \mathcal{A}$.

Example 2.14. Define $\mathcal{C} = \mathcal{A}^T * \mathcal{A}$. We can compute this via n matrix products in the Fourier domain. But, due to the tensor transpose definition and conjugate symmetry of Fourier coefficients, this means $\hat{C}^{(i)} = (\hat{A}^{(i)})^H (\hat{A}^{(i)})$. That is, each face of $\hat{\mathcal{C}}$ is Hermitian and at least semidefinite. Also we only have to do about half of the facewise multiplications to calculate $\mathcal{C}$.

This example motivates a new definition.

DEFINITION 2.15. $\mathcal{A} \in \mathbb{K}_n^{m\times m}$ *is* symmetric positive definite *if the* $\hat{A}^{(i)}$ *are Hermitian positive definite.*

Clearly, $\mathcal{C}$ as defined in the previous example is symmetric positive definite.

Let us now investigate what the definitions in section 2 imply when specifically applied to elements of $\mathbb{K}_n^m$ (which, according to the previous discussion, we can think of interchangeably as elements of $\mathbb{R}^{m\times n}$).

3. Inner products, norms, and orthogonality of matrices. In the following, uppercase letters always describe the matrix equivalent of an $m \times 1 \times n$ tensor. That is, in terms of our previous definitions, $X := \texttt{squeeze}(\vec{\mathcal{X}})$.

Suppose that $\vec{\mathcal{X}}, \vec{\mathcal{Y}} \in \mathbb{K}_n^m$. Then the authors of [16] refer to $\mathbf{a} := \vec{\mathcal{X}}^T * \vec{\mathcal{Y}} \in \mathbb{K}_n$ as an *inside product* in analogy with the notion of an inner product. Indeed, with the appropriate definition of "conjugacy" we can use this observation to determine a bilinear form on $\mathbb{K}_n^m$.

LEMMA 3.1. *Let* $\vec{\mathcal{X}}, \vec{\mathcal{Y}}, \vec{\mathcal{Z}} \in \mathbb{K}_n^m$ *and let* $\mathbf{a} \in \mathbb{K}_n$. *Then* $\langle \vec{\mathcal{X}}, \vec{\mathcal{Y}} \rangle := \vec{\mathcal{X}}^T * \vec{\mathcal{Y}}$ *satisfies the following:*

1. $\langle \vec{\mathcal{X}}, \vec{\mathcal{Y}} + \vec{\mathcal{Z}} \rangle = \langle \vec{\mathcal{X}}, \vec{\mathcal{Y}} \rangle + \langle \vec{\mathcal{X}}, \vec{\mathcal{Z}} \rangle$.
2. $\langle \vec{\mathcal{X}}, \vec{\mathcal{Y}} * \mathbf{a} \rangle = (\vec{\mathcal{X}}^T * \vec{\mathcal{Y}}) * \mathbf{a} = \mathbf{a} * (\vec{\mathcal{X}}^T * \vec{\mathcal{Y}}) = \mathbf{a} * \langle \vec{\mathcal{X}}, \vec{\mathcal{Y}} \rangle$.
3. $\langle \vec{\mathcal{X}}, \vec{\mathcal{Y}} \rangle = \langle \vec{\mathcal{Y}}, \vec{\mathcal{X}} \rangle^T$.

Of course, $\langle\ ,\ \rangle$ does not produce a map to the reals, and so in the traditional sense does not define an inner product. Indeed, $\langle \vec{\mathcal{X}}, \vec{\mathcal{X}} \rangle$ is a $1 \times 1 \times n$ tensor and it is possible for $\langle \vec{\mathcal{X}}, \vec{\mathcal{X}} \rangle$ to have negative entries, so the standard notion of positivity of the bilinear form does not hold.

On the other hand, the (1,1,1) entry in tubal scalar $\langle \vec{\mathcal{X}}, \vec{\mathcal{X}} \rangle$, denoted $\langle \vec{\mathcal{X}}, \vec{\mathcal{X}} \rangle^{(1)}$, is given by $\texttt{vec}(X)^T\texttt{vec}(X)$. In other words, $\langle \vec{\mathcal{X}}, \vec{\mathcal{X}} \rangle^{(1)}$ is the square of the Frobenius norm of $\vec{\mathcal{X}}$ (likewise of X). Thus, it is zero only when $\vec{\mathcal{X}}$ is 0, and nonnegative otherwise. Therefore, we can make the following definition.

DEFINITION 3.2. *Given* $\vec{\mathcal{X}} \neq 0 \in \mathbb{K}_n^m$ *and* $\vec{\mathcal{X}} = \texttt{twist}(X)$, *the length of any nonzero* $\vec{\mathcal{X}}$ *is given as*

$$\|\vec{\mathcal{X}}\| := \frac{\|\langle \vec{\mathcal{X}}, \vec{\mathcal{X}} \rangle\|_F}{\sqrt{\langle \vec{\mathcal{X}}, \vec{\mathcal{X}} \rangle^{(1)}}} = \frac{\|\langle \vec{\mathcal{X}}, \vec{\mathcal{X}} \rangle\|_F}{\|\vec{\mathcal{X}}\|_F},$$

while $\|\vec{\mathcal{X}}\| = 0$ *if* $\vec{\mathcal{X}} = 0$.

Note that $\vec{\mathcal{X}}$ can only have unit length if $\langle \vec{\mathcal{X}}, \vec{\mathcal{X}} \rangle = \mathbf{e}_1$. Note also that when $n = 1$, so that $\vec{\mathcal{X}}$ is in $\mathbb{R}^m$, this definition coincides with the 2-norm of that vector.

For completeness, we define a tube of angles between two matrices. More detailed discussion and an intuitive illustration are reserved for Appendix B.

DEFINITION 3.3. *Given nonzero* $\vec{\mathcal{X}}, \vec{\mathcal{Y}} \in \mathbb{K}_n^m$, *the tubal angle between them is given implicitly by the tube*

$$\cos(\boldsymbol{\theta}) = \frac{1}{2\|\vec{\mathcal{X}}\|_F\|\vec{\mathcal{Y}}\|_F}(|\langle \vec{\mathcal{X}}, \vec{\mathcal{Y}} \rangle + \langle \vec{\mathcal{Y}}, \vec{\mathcal{X}} \rangle|),$$

where $|\cdot|$ *is understood to be componentwise absolute value.*

Note that when $n = 1$, so that $\vec{\mathcal{X}}, \vec{\mathcal{Y}}$ are in $\mathbb{R}^m$, this definition coincides with the usual notion of angle between two vectors. Otherwise, we have a length-n tube of angles (i.e., a length-n vector) describing the orientation of one matrix relative to the another.

Using the bilinear form, the definition of orthogonal tensors in Definition 2.9 and Observation 2.11, we are in a position to define what we mean by an orthogonal set[2] in $\mathbb{K}_n^m$.

DEFINITION 3.4. *Given a collection of k, $m \times n$ matrices X_j, with corresponding tensors $\vec{\mathcal{X}}_j = \texttt{twist}(X_j)$, the collection is called* orthogonal *if*

$$\langle \vec{\mathcal{X}}_i, \vec{\mathcal{X}}_j \rangle = \begin{cases} \alpha_i \mathbf{e}_1, & i = j, \\ \mathbf{0}, & i \neq j, \end{cases}$$

where α_i is a nonzero scalar. The set is orthonormal *if $\alpha_i = 1$.*

Suppose $\mathcal{Q}$ is an orthogonal tensor. It is easily verified that the (i, j) tubal scalar entry of the product $\mathcal{Q}^T * \mathcal{Q}$ is given by $\vec{\mathcal{Q}}_i^T * \vec{\mathcal{Q}}_j = \langle \vec{\mathcal{Q}}_i, \vec{\mathcal{Q}}_j \rangle$, and since $\mathcal{Q}$ is orthogonal, the (i, j) entry is $\mathbf{e}_1$ if $i = j$ and $\mathbf{0}$ otherwise. Thus this definition of orthogonality is consistent with our earlier definition of an orthogonality for tensors in that $\mathcal{Q} \in \mathbb{R}^{m \times m \times n}$ is an orthogonal tensor iff the lateral slices $\{\vec{\mathcal{Q}}_1, \vec{\mathcal{Q}}_2, \ldots, \vec{\mathcal{Q}}_m\}$ form an orthonormal set.

4. Linear combinations with tubal scalars, range, and kernel. In the previous section, we gave a definition of a set of orthogonal matrices in $\mathbb{K}_n^m$. In analogy with standard linear algebra and in light of the framework we have set up, one would hope that if the orthogonal set contains m elements, we should be able to reconstruct any element in $\mathbb{K}_n^m$ from those m elements. That is obviously *not* the case if we consider "linear combination" in the traditional sense, with the scalars as elements of $\mathbb{R}$. However, it *will* be the case if we consider what we will call t-linear combinations, where tubal-scalars now take the role of scalars.

DEFINITION 4.1. *Given k tubal scalars $\mathbf{c}_j, j = 1, \ldots, k$, in $\mathbb{K}_n$, a t-linear combination of $\vec{\mathcal{X}}_j, j = 1, \ldots, k$, of $\mathbb{K}_n^m$ is defined as*

$$\vec{\mathcal{X}}_1 * \mathbf{c}_1 + \vec{\mathcal{X}}_2 * \mathbf{c}_2 + \cdots + \vec{\mathcal{X}}_k * \mathbf{c}_k \equiv \mathcal{X} * \vec{\mathcal{C}}, \text{ where } \mathcal{X} := [\vec{\mathcal{X}}_1, \ldots, \vec{\mathcal{X}}_k], \ \vec{\mathcal{C}} := \begin{bmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_k \end{bmatrix}.$$

Note that the order of multiplication is important, as in general $\mathbf{c}_j * \vec{\mathcal{X}}_j$ will not be defined unless $m = 1$.

Example 4.2. Using $\mathcal{A} \in \mathbb{K}_2^{3\times 2}$ and $\vec{\mathcal{B}} \in \mathbb{K}_2^2$ from Example 2.6, we see that

$$\begin{aligned} \mathcal{A} * \vec{\mathcal{B}} &= \vec{\mathcal{A}}_1 * \mathbf{b}_{11} + \vec{\mathcal{A}}_2 * \mathbf{b}_{21} \\ &= \texttt{fold}\left(\left[\begin{array}{c} 7 \\ 4 \\ -3 \\ \hline -8 \\ -6 \\ 2 \end{array}\right]\right) + \texttt{fold}\left(\left[\begin{array}{c} -3 \\ -23 \\ 0 \\ \hline -1 \\ -13 \\ -8 \end{array}\right]\right) = \texttt{fold}\left(\left[\begin{array}{c} 4 \\ -19 \\ -3 \\ \hline -9 \\ -19 \\ -6 \end{array}\right]\right). \end{aligned}$$

Thus, $\vec{\mathcal{C}} := \mathcal{A} * \vec{\mathcal{B}}$ is a t-linear combination of the lateral slices of $\mathcal{A}$.

Next, observe that a matrix Y can be built from the matrices $\texttt{squeeze}(\vec{\mathcal{X}}_j)$ as follows.

[2] We could equivalently refer to this as an orthogonal set of matrices in $\mathbb{R}^{m\times n}$, but tend to avoid this nomenclature since it could be confused with a set of orthogonal matrices, which is different.

LEMMA 4.3. *With $\mathcal{X}$, $\vec{\mathcal{C}}$ as defined in Definition* 4.1*,*

$$Y := \texttt{squeeze}(\mathcal{X} * \vec{\mathcal{C}}) = \sum_{i=1}^{k} X_i \texttt{bcirc}(\mathbf{c}_i^T) \in \mathbb{R}^{m\times n}. \tag{4.1}$$

Proof. The proof follows from (2.1) and from $\texttt{squeeze}(\vec{\mathcal{X}} + \vec{\mathcal{Y}}) = \texttt{squeeze}(\vec{\mathcal{X}}) + \texttt{squeeze}(\vec{\mathcal{Y}})$. ∎

Since the circulant matrices $\texttt{bcirc}(\mathbf{c}_i^T)$ will not be scalar multiples of the identity in general, this isn't a linear combination of the X_i. However, since these $n \times n$ circulant matrices can be diagonalized by the $n \times n$ DFT matrix, it follows that the jth column of the matrix YF_n is a (traditional) linear combination of the jth columns of X_iF_n, $i = 1, \ldots, k$. This multilinear perspective will be helpful in interpreting the range of the t-linear operator in the next subsection.

4.1. Tubal rank, range, and kernel. There is one important sense in which tubal scalars differ from elements in $\mathbb{R}$. Even though $\mathbf{a}$ may have all nonzero entries, $\mathbf{a}$ may not be invertible.[3] According to the definition of invertibility, $\mathbf{a}$ is only invertible if there exists $\mathbf{b}$ such that $\mathbf{a} * \mathbf{b} = \mathbf{b} * \mathbf{a} = \mathbf{e}_1$. However, we noted previously that t-products can be implemented by Fourier transforming in the third dimension, computing pairwise products of faces, then inverse Fourier transforming the result. Thus, an element $\mathbf{a} \in \mathbb{K}_n$ can be invertible iff $\hat{\mathbf{a}}$ (obtained by Fourier transforming $\mathbf{a}$ in the third dimension) has no zero entries (i.e., $\mathbf{a}$ can have no zero Fourier coefficients).

In order to appropriately characterize the range, denoted $R(\mathcal{A})$, and kernel, denoted $N(\mathcal{A})$, of the map defined by tensor $\mathcal{A}$, it is necessary to capture information about the dimensionality that is more subtle than in the standard linear algebra situation of matrices over a scalar field. Specifically, we need to separate tubal scalars that are invertible from those that are not.

DEFINITION 4.4. *Suppose $\mathbf{b} \in \mathbb{K}_n$. Then its* tubal rank *is the number of its nonzero Fourier coefficients. If its tubal rank is n, we say it is* invertible*; if it is less than n, it is not.*[4] *In particular, the tubal rank is* 0 *iff $\mathbf{b} = \mathbf{0}$.*

We will use the tensor SVD, or t-SVD introduced in [16], to characterize $R(\mathcal{A})$ and $N(\mathcal{A})$ for $\mathcal{A} \in \mathbb{R}^{\ell\times m\times n}$. The authors show there exists an $\ell \times \ell \times n$ orthogonal $\mathcal{U}$, an $\ell \times m \times n$ f-diagonal $\mathcal{S}$, and an $m \times m \times n$ orthogonal $\mathcal{V}$ such that

$$\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T = \sum_{i=1}^{\min(\ell,m)} \vec{\mathcal{U}}_i * \mathbf{s}_i * \vec{\mathcal{V}}_i^T, \qquad \mathbf{s}_i := \mathcal{S}(i,i,:),$$

where the $\mathbf{s}_i$ are the singular tuples. (See Figure 4.1.)

It is worth noting that the t-SVD can be derived using (2.2). Specifically, the tensors $\mathcal{U}, \mathcal{S}, \mathcal{V}$ are derived from individual matrix SVDs in Fourier space; that is, $\hat{U}^{(i)}\hat{S}^{(i)}(\hat{V}^{(i)})^T = \hat{A}^{(i)}$. If $\hat{A}^{(i)}$ has rank $r_i < \min(\ell, m)$, then for $j \geq r_i$, $\hat{\mathbf{s}}_j$ has a 0 in the ith position; i.e., $\mathbf{s}_j$ has at least one zero Fourier coefficient, and therefore $\mathbf{s}_j$ is not invertible. If all the faces of $\hat{\mathcal{A}}$ have ranks less than $\min(\ell, m)$, then there will be at least one $\mathbf{s}_j$ which is identically $\mathbf{0}$ because all n of its Fourier coefficients are zero. However, we will need a way of keeping track of the $\mathbf{s}_j$ that have zero Fourier coefficients but which are not necessarily equal to $\mathbf{0}$ if we want to specify something useful about the range and/or kernel.

[3] This is equivalent to the observation that $\mathbb{K}_n$ in [1] cannot form a field.

[4] These observations are consistent with those in [16] adapted to the special case when $\ell = m = 1$.

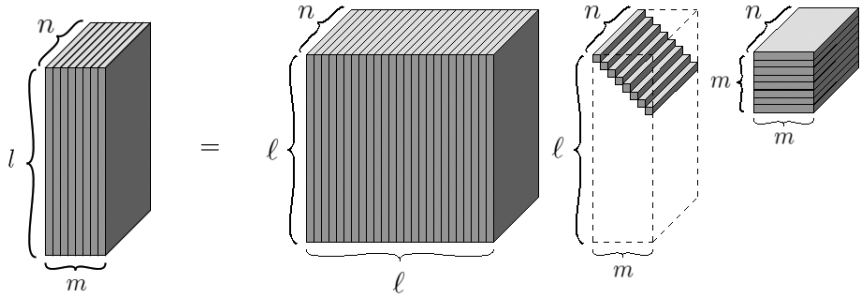

FIG. 4.1. *The t-SVD of an $l \times m \times n$ tensor. (Originally appeared in* [11]*.)*

Let $p = \min(\ell, m)$. Ideally, we would like to be able to describe the range of $\mathcal{A}$ in terms of a t-linear combination of a subset of the $\vec{\mathcal{U}}_i$, and the kernel in terms of a subset of the $\vec{\mathcal{V}}_i$. As we will see, this will be sufficient in the case when all the nonzero $\mathbf{s}_i$ are invertible. But when there are some nonzero $\mathbf{s}_i$ which are not invertible, we will have to take special care. Indeed, observe that for any $\vec{\mathcal{X}} \in \mathbb{K}_n^m$,

$$\mathcal{A} * \vec{\mathcal{X}} = \sum_{i=1}^{\min(\ell,m)} \vec{\mathcal{U}}_i * (\mathbf{s}_i * \underbrace{\vec{\mathcal{V}}_i^T * \vec{\mathcal{X}}}_{\mathbf{d}_i}), \in \mathbb{K}_n^{\ell}, \tag{4.2}$$

where the term in parenthesis is a tubal scalar that is the product of two tubal scalars, $\mathbf{s}_i$ and $\mathbf{d}_i$. Hence, the range of $\mathcal{A}$ is a t-linear combination of the $\vec{\mathcal{U}}_i$. However, the range is special because *each term in parentheses will have zero Fourier coefficients exactly where each* $\mathbf{s}_i$ *has zero Fourier coefficients.* To see this more clearly, observe that from Lemma 4.3, in Fourier space is

$$\widehat{\texttt{squeeze}(\mathcal{A} * \vec{\mathcal{X}})} = \sum_{i=1}^{p} [\hat{\mathbf{s}}_i^{(1)} \hat{\mathbf{d}}_i^{(1)} \hat{U}^{(1)}(:,i), \hat{\mathbf{s}}_i^{(2)} \hat{\mathbf{d}}_i^{(2)} \hat{U}^{(2)}(:,i), \ldots, \hat{\mathbf{s}}_i^{(n)} \hat{\mathbf{d}}_i^{(n)} \hat{U}^{(n)}(:,i)], \tag{4.3}$$

where, as mentioned above, $\hat{U}^{(i)} \hat{S}^{(i)} (\hat{V}^{(i)})^T = \hat{A}^{(i)}$. In summary, the jth column of $\widehat{\texttt{squeeze}(\mathcal{A} * \vec{\mathcal{X}})}$ is a linear combination of the first p columns of $\hat{U}^{(j)}$ with scalar expansion coefficients $\hat{\mathbf{s}}_i^{(j)} \hat{\mathbf{d}}_i^{(j)}, i = 1, \ldots, p$.

Accordingly, we define the following concept.

DEFINITION 4.5. *The* multirank *of the $\mathcal{A}$ is the tubal scalar $\boldsymbol{\rho} := \boldsymbol{\rho}(\mathcal{A}) \in \mathbb{K}_n$ such that $\boldsymbol{\rho}^{(i)}$ is the rank of the ith matrix $\hat{A}^{(i)}$. Since $\hat{A}^{(i)}$ is $\ell \times m$, $\boldsymbol{\rho}(i) \leq \min(\ell, m)$. The* multinullity *is the complimentary tubal scalar $\boldsymbol{\eta} := \boldsymbol{\eta}(\mathcal{A})$ with entries $m - \boldsymbol{\rho}^{(i)}$.*

Next, we characterize range and null space.

THEOREM 4.6. *Suppose the first j singular tuples are invertible, the next k are nonzero but not invertible, and that for $p = \min(\ell, m)$, $p - (j + k)$ are identically $\mathbf{0}$. The sets $R(\mathcal{A}), N(\mathcal{A})$ are given unambiguously by*

$$R(\mathcal{A}) = \{\vec{\mathcal{U}}_1 * \mathbf{c}_1 + \cdots + \vec{\mathcal{U}}_{j+k} * \mathbf{c}_{j+k} | \mathbf{c}_i = \mathbf{s}_i * \mathbf{d}_i, \mathbf{d}_i \in \mathbb{K}_n, j < i \leq j + k\},$$

$$N(\mathcal{A}) = \{\vec{\mathcal{V}}_{j+1} * \mathbf{c}_{j+1} + \cdots + \cdots + \vec{\mathcal{V}}_m * \mathbf{c}_m | \mathbf{s}_i * \mathbf{c}_i = \mathbf{0}, j < i \leq j + k\}.$$

Proof. That anything in the range must be a t-linear combination of the first $j+k$ lateral slices of $\mathcal{U}$ is clear from (4.2). However, if there are singular tuples with nonzero tubal rank less than n, it is not true that every t-linear combination of the of the first $j+k$ lateral slices of $\mathcal{U}$ is in the range.

Indeed, the equation

$$\mathcal{A} * \vec{\mathcal{X}} = \sum_{i=1}^{k+j} \vec{\mathcal{U}}_i * \mathbf{c}_i$$

is solvable only if $\mathbf{c}_i, i = j+1, \ldots, j+k$, has zero Fourier coefficients in exactly the same positions as the zero Fourier coefficients of $\mathbf{s}_i$ for those terms (i.e., compare the right-hand side of (4.3) to the right-hand side of the Fourier-space version of the above equation). In other words, we need $\mathbf{c}_i = \mathbf{d}_i * \mathbf{s}_i$ for some $\mathbf{d}_i$ for $i = j+1, \ldots, j+k$.

Now, if $j+k+1 \leq m$,

$$\mathcal{A} * \vec{\mathcal{V}}_i = \mathbf{0}, \qquad i = j+k+1, \ldots, m,$$

but there are other vectors that map to zero as well. For example, let $\mathbf{c}_i$ be a tubal scalar that has zero Fourier coefficients where $\mathbf{s}_i$, for $j+1 \leq i \leq j+k$, has nonzero Fourier coefficients, and 1's where $\mathbf{s}_i$ has zero Fourier coefficients. Then

$$\mathcal{A} * (\vec{\mathcal{V}}_i * \mathbf{c}_i) = \mathbf{0} \qquad \text{but} \qquad \mathcal{A} * \vec{\mathcal{V}}_i \neq \mathbf{0}.$$

With these observations, the result is proved. ∎

This analysis shows that the *number of matrices (elements of $\mathbb{K}_n^m$) necessary to generate any element in $R(\mathcal{A})$ is equal to the number of nonzero singular tuples, whether or not those tuples are invertible.* The number of matrices necessary to generate an arbitrary element of $N(\mathcal{A})$ is equal to $m-p$ plus the number of nonzero and noninvertible singular tuples. For this reason, we refer to the first $j+k$ columns of $\mathcal{U}$ as the *generators* for the range (but abstain from using the term basis, in the sense that knowledge of the tubal ranks of the singular tuples must be included to describe the range). Likewise, columns $j+1$ through m of $\mathcal{V}$ are the *generators* for the null space.

Furthermore, a traditional rank plus nullity theorem does not hold if we consider the "dimension" of $R(\mathcal{A})$ as the min/max number of matrices necessary to generate any element in the set, and similarly for $N(\mathcal{A})$. Thus, when we define dimension, it should be consistent with the multi-rank of the matrix (see also (4.3)).

Thus, we set $\dim(R(\mathcal{A})) := \|\boldsymbol{\rho}(\mathcal{A})\|_1$, $\dim(N(\mathcal{A})) := \|\boldsymbol{\eta}(\mathcal{A})\|_1$, and under this convention, it will always be the case that

$$\dim(R(\mathcal{A})) + \dim(N(\mathcal{A})) = nm = \|\boldsymbol{\rho}(\mathcal{A}) + \boldsymbol{\eta}(\mathcal{A})\|_1.$$

Finally, we set a notion of conditioning for a tensor that is consistent with the aforementioned new concepts.

DEFINITION 4.7. *If $\mathcal{A} \in \mathbb{R}^{\ell \times m \times n}$, its condition number is infinite if $m > \ell$ or the multinullity is nonzero (i.e., $\|\boldsymbol{\eta}(\mathcal{A})\|_1 > 0$). If the condition number is not infinite, then the* condition tubal scalar *is well defined and is given by the length n tubal scalar with entries defined by the condition numbers of each of the respective $\hat{A}^{(i)}$.*

5. Orthogonal projectors and Gram–Schmidt for orthogonal matrices. Now that we understand how to define the range and kernel in terms of t-linear

combinations, you might ask whether it is possible to design orthogonal projectors onto the spaces.

The following definition is consistent with the concept of projectors with respect to matrices.

DEFINITION 5.1. *$\mathcal{P}$ is a projector if $\mathcal{P}^2 = \mathcal{P} * \mathcal{P} = \mathcal{P}$, and it is orthogonal if $\mathcal{P}^T = \mathcal{P}$.*

In particular, if $\{\vec{\mathcal{Q}}_i\}_{i=1}^k$ is an orthogonal set in $\mathbb{K}_n^m$, then $\mathcal{P} = [\vec{\mathcal{Q}}_1, \ldots, \vec{\mathcal{Q}}_k] * [\vec{\mathcal{Q}}_1, \ldots, \vec{\mathcal{Q}}_k]^T$ defines an orthogonal projector, as does $\mathcal{I} - \mathcal{P}$.

Note that when $\mathcal{A}^T * \mathcal{A}$ is invertible, $\mathcal{A} * (\mathcal{A}^T * \mathcal{A})^{-1} * \mathcal{A}^T$ is an orthogonal projector onto $R(\mathcal{A})$. If one or more singular tuples are not invertible, then to describe an orthogonal projector onto $R(\mathcal{A})$ requires more information than just an orthogonal set of generating matrices for the $R(\mathcal{A})$, as noted in Theorem 4.6. Define the diagonal tensor $\mathcal{D}$ so that the Fourier coefficients of each tubal scalar along the diagonal satisfy

$$\hat{\mathbf{d}}_i^{(j)} = \begin{cases} 1 & \text{if } \hat{\mathbf{s}}_i^{(j)} \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

It is easily shown that an orthogonal projector onto $R(\mathcal{A})$ is given by $\mathcal{U}_k * \mathcal{D} * \mathcal{U}_k^T$, where k is the number of nonzero singular tuples and $\mathcal{U}_k = \mathcal{U}(:, 1\!:\!k, :)$.

It is natural to think about a classical Gram–Schmidt process for generating an orthonormal set of matrices. The key, however, to doing it correctly is the normalization. Algorithm 2 takes a nonzero $\vec{\mathcal{X}} \in \mathbb{K}_n^m$ and returns $\vec{\mathcal{V}} \in \mathbb{K}_n^m$, $\mathbf{a} \in \mathbb{K}_n$ such that

$$\vec{\mathcal{X}} = \vec{\mathcal{V}} * \mathbf{a},$$

where $\|\vec{\mathcal{V}}\| = 1$. Note that $\mathbf{a}$ *might not be invertible*. Recall that $\mathbf{a}^{(j)}$ is a scalar (i.e., the jth frontal face of the $1 \times 1 \times n$ tensor $\mathbf{a}$), and that if $\vec{\mathcal{X}}$ is $m \times 1 \times n$, $\vec{\mathcal{X}}^{(j)}$ is the jth frontal face of $\vec{\mathcal{X}}$, which is vector of length m.

The following classical Gram–Schmidt algorithm (see Algorithm 3) takes $\mathcal{A} \in \mathbb{R}^{\ell \times m \times n}$ with $\ell \geq m$ as input and returns the factorization $\mathcal{A} = \mathcal{Q} * \mathcal{R}$, where $\mathcal{Q}$ is $\ell \times m \times n$ with orthonormal "columns" $\vec{\mathcal{Q}}_k$, $k = 1, \ldots, m$, and $\mathcal{R}$ is $m \times m \times n$ f-upper triangular. The tubes on the diagonal of $\mathcal{R}$ correspond to normalization terms.

Algorithm 2: Normalize.

Input: $\vec{\mathcal{X}} \in \mathbb{K}_n^m \neq 0$
Output: $\vec{\mathcal{V}} * \mathbf{a} = \vec{\mathcal{X}}$ with $\|\vec{\mathcal{V}}\| = 1$

$\vec{\mathcal{V}} \leftarrow \texttt{fft}(\vec{\mathcal{X}}, [\,], 3)$
for $j = 1$ **to** n
  $\mathbf{a}^{(j)} \leftarrow \|\vec{\mathcal{V}}^{(j)}\|_2$ (note $\vec{\mathcal{V}}^{(j)}$ is a vector)
  if $\mathbf{a}^{(j)} > tol$ **then**
    $\vec{\mathcal{V}}^{(j)} \leftarrow \frac{1}{\mathbf{a}^{(j)}} \vec{\mathcal{V}}^{(j)}$
  else
    $\vec{\mathcal{V}}^{(j)} \leftarrow \texttt{randn}(n, 1)$; $\mathbf{a}^{(j)} \leftarrow \|\vec{\mathcal{V}}^{(j)}\|_2$; $\vec{\mathcal{V}}^{(j)} \leftarrow \frac{1}{\mathbf{a}^{(j)}} \vec{\mathcal{V}}^{(j)}$; $\mathbf{a}^{(j)} \leftarrow 0$
  end
  $\vec{\mathcal{V}} \leftarrow \texttt{ifft}(\vec{\mathcal{V}}, [\,], 3)$; $\mathbf{a} \leftarrow \texttt{ifft}(\mathbf{a}, [\,], 3)$
end for

Algorithm 3: Classical Gram–Schmidt, QR version.

Input: $\mathcal{A} \in \mathbb{K}_n^{\ell \times p}$, $\ell \geq p$
Output: $\mathcal{Q} \in \mathbb{K}_n^{\ell \times p}, \mathcal{R} \in \mathbb{K}_n^{p \times p}$ such that $\mathcal{A} = \mathcal{Q} * \mathcal{R}$

$[\vec{\mathcal{Q}}_1, \mathcal{R}(1,1,:)] \leftarrow \text{Normalize}(\vec{\mathcal{A}}_1)$
for $i = 2$ **to** m
  $\vec{\mathcal{X}} \leftarrow \vec{\mathcal{A}}_i$;
  for $j = 1$ **to** $i - 1$
    $\mathcal{R}(j,i,:) \leftarrow \vec{\mathcal{Q}}_j^T * \vec{\mathcal{X}}$;
    $\vec{\mathcal{X}} \leftarrow \vec{\mathcal{X}} - \vec{\mathcal{Q}}_j * \mathcal{R}(j,i,:)$;
  end for
  $[\vec{\mathcal{Q}}_i, \mathcal{R}(i,i,:)] \leftarrow \text{Normalize}(\vec{\mathcal{X}})$;
end for

This algorithm is consistent with the tensor QR factorization introduced in [17, 16] and described in the first section. Having introduced the concept of orthogonal projectors, it is straightforward to derive the modified Gram–Schmidt analogue of this, so we will leave this exercise to the reader. If the number of nonzero tubal scalars on the diagonal of $\mathcal{R}$ is k, then the first k lateral slices of $\mathcal{Q}$ can be used to describe (or project onto) $R(\mathcal{A})$; however, as described above, if some of the diagonals of $\mathcal{R}$ are not invertible, this information must be taken into account when generating (or projecting onto) the range.

6. Tensor polynomials, computation of tensor eigendecomposition, and Krylov subspace methods. In this section, we will assume $\hat{\mathcal{A}}$ has diagonalizable faces, that is, $\hat{\mathcal{A}}^{(i)} = \hat{X}^{(i)}\hat{D}^{(i)}(\hat{X}^{(i)})^{-1}$. It follows that if the invertible tensor $\hat{\mathcal{X}}$ and f-diagonal tensor $\hat{\mathcal{D}}$ are defined facewise as $\hat{\mathcal{X}}^{(i)} = \hat{X}^{(i)}$ and $\hat{\mathcal{D}}^{(i)} = \hat{D}^{(i)}$, then, upon taking the inverse FFT along tubes, we have an eigendecomposition [1]

$$\mathcal{A} = \mathcal{X} * \mathcal{D} * \mathcal{X}^{-1} \Longrightarrow \mathcal{A} * \mathcal{X} = \mathcal{X} * \mathcal{D} \Longrightarrow \mathcal{A} * \vec{\mathcal{X}}_j = \vec{\mathcal{X}}_j * \mathbf{d}_j,$$

where the latter follows from Observation 2.11. See [9] for a definition of a canonical decomposition and explicit ordering of eigentuples in a slightly different notational framework.

A word of caution to the reader. Eigenvalue decomposition for tensors means different things to different researchers (see [19, 22, 24, 25], for example). In some scenarios, one really does desire a single scalar eigenvalue and an eigenvector of length n. Our approach differs in that we are looking for *eigentuples* $\mathbf{d}_j$ and their corresponding eigenmatrices $\vec{\mathcal{X}}_j$. In our case, eigendecompositions can exist even when $m \neq n$, although $\ell = m$ is required.

Recall from the first section that the faces of $\hat{\mathcal{A}} \equiv \hat{\mathcal{B}^T} * \hat{\mathcal{B}}$ are Hermitian semi-definite. Thus, if $\mathcal{A} = \mathcal{B}^T * \mathcal{B}$ for some tensor $\mathcal{B}$, such an eigendecomposition exists. It has been shown in [16, 2, 10] that the t-SVD is useful in image processing applications. In analogy with the matrix case, it is easy to show that there is a connection between the t-SVD of $\mathcal{B}$ and the eigendecompositions of $\mathcal{B}^T * \mathcal{B}$ and $\mathcal{B} * \mathcal{B}^T$. This suggests that efficient algorithms to compute eigenpairs can be used to compute full or partial t-SVDs.

The concept of determinant for third-order tensors was defined in [15, 9], its implication in terms of the number of eigentuples for $\mathcal{A} \in \mathbb{K}_n^{n \times n}$ (up to $\binom{n/2+1}{2}$ for

n even, for instance), was also cited in those works, and will not be pursued further here. The upshot of the results in those papers is that, in light of (2.2), computing eigentuples and corresponding eigenmatrices is equivalent to computing eigenvalues and eigenvectors for the blocks $\hat{A}^{(i)}$ simultaneously, but *choosing a consistent ordering scheme* for the individual matrix eigendecompositions is crucial.

Moreover, as we will see in the next section, *generalizing matrix eigenvalue algorithms, such as the power iteration, and Krylov subspace iteration will not be completely straightforward* without appropriate normalization.

6.1. Powers of tensors. If we want to investigate the extension of any matrix algorithm based implicitly on powers of matrices (e.g., the power iteration and Krylov subspace methods) to tensors under the t-product framework, we need to look closely at what $\mathcal{A}^k$ means for positive integer k and our $m \times m \times n$ tensor, $\mathcal{A}$.

Let k be a positive integer. Then $\mathcal{A}^k := \overbrace{\mathcal{A} * \mathcal{A} \cdots * \mathcal{A}}^{k \text{ times}}$ can be computed by

1. $\hat{\mathcal{A}} = \texttt{fft}(\mathcal{A}, [\,], 3)$;
2. for $i = 1 : n$
   $\hat{\mathcal{C}}^{(i)} = (\hat{\mathcal{A}}^{(i)})^k$ (a matrix power);
3. $\mathcal{A}^k = \texttt{ifft}(\hat{\mathcal{C}}, [\,], 3)$.

Recall our assumption $\hat{A}^{(i)} = \hat{X}^{(i)}\hat{D}^{(i)}(\hat{X}^{(i)})^{-1}$. Then $(\hat{A}^{(i)})^k = \hat{X}^{(i)}(\hat{D}^{(i)})^k(\hat{X}^{(i)})^{-1}$. For convenience, suppose entries in each of the diagonal matrices $\hat{D}^{(i)}$ are ordered in decreasing magnitude so that the (1,1) entry of $\Lambda^{(i)}$, denoted as $\lambda_1^{(i)}$, has the property $|\lambda_1^{(i)}| > |\lambda_2^{(i)}| \geq \cdots \geq |\lambda_m^{(i)}|$.

Let $\vec{\mathcal{W}} \in \mathbb{K}_n^m$ be the initial vector for a power iteration. Consider $\mathcal{A}^k * \vec{\mathcal{W}}$. Now $\vec{\mathcal{W}} = \sum_{i=1}^m \vec{\mathcal{V}}_i * \mathbf{a}_i$ for some tubal scalars $\mathbf{a}_i$, and so

$$\mathcal{A}^k * \vec{\mathcal{W}} = \sum_{i=1}^m \vec{\mathcal{V}}_i * \mathbf{d}_i^k * \mathbf{a}_i = \mathcal{D}[\mathbf{d}_1]^k \sum_{i=1}^m \vec{\mathcal{V}}_i * \left(\frac{\mathbf{d}_i}{\mathbf{d}_1}\right)^k * \mathbf{a}_i,$$

where $\mathcal{D}[\mathbf{d}_1]$ is the diagonal tensor with tubal scalar $\mathbf{d}_1$ repeated along the diagonal.

So, one would expect that, *under this ordering of eigenvalues in the Fourier domain*, a power iteration for the tensor $\mathcal{A}$ (in the original space) should converge to the eigentuple corresponding to the largest magnitude entries of each $\hat{D}^{(i)}$. But without the proper normalization of the candidate eigenmatrix at each iteration, this will not happen! The easiest way to see this is to recall (2.2) so that we can either try to do a power iteration on the big block diagonal matrix on the right, or we can do individual power iterations on each of the blocks. What we want to do is equivalent to running individual power iterations on each of the blocks, which requires that each of the individual length-m candidate eigenvectors be normalized (in the 2-norm) to length 1. But without this normalization, one would end up running the power iteration on the big block diagonal matrix, which would pick off only one (or 2, given conjugate symmetry) eigenvalues of the big block diagonal matrix and their corresponding length-m eigenvectors.

We therefore propose the following power iteration (see Algorithm 4), which uses our notation of length and normalization from the previous section, which will converge to the eigentuple that corresponds to finding the largest magnitude eigenvalue of each $\hat{A}^{(i)}$.

We conclude this section by noting that extending algorithms such as the Arnoldi iteration [9] and QR iteration [15, 3] are likewise possible, though a number of numerical issues must be addressed in order for these methods to be properly implemented [9].

Algorithm 4: Power iteration.

Input: $\mathcal{A} \in \mathbb{K}_n^{m\times m}$
Output: $\mathbf{d} \in \mathbb{K}_n, \vec{\mathcal{V}} \in \mathbb{K}_n^m$ such that $\mathcal{A} * \vec{\mathcal{V}} \approx \vec{\mathcal{V}} * \mathbf{d}$

$\vec{\mathcal{V}} \leftarrow \texttt{randn}(m, 1, n)$;
$\vec{\mathcal{V}} \leftarrow \text{Normalize}(\vec{\mathcal{V}})$;
for $i = 1$ **to** ...
  $\vec{\mathcal{V}} \leftarrow \mathcal{A} * \vec{\mathcal{V}}$;
  $\vec{\mathcal{V}} \leftarrow \text{Normalize}(\vec{\mathcal{V}})$;
  $\mathbf{d} \leftarrow \vec{\mathcal{V}}^T * (\mathcal{A} * \vec{\mathcal{V}})$;
end for

6.2. Krylov subspace methods. The Krylov tensor generated by $\mathcal{G}$ and $\vec{\mathcal{B}}$ composed of k lateral slices is defined according to

$$\mathcal{K}_k(\mathcal{G}, \vec{\mathcal{B}}) := [\vec{\mathcal{B}}, \mathcal{G} * \vec{\mathcal{B}}, \mathcal{G}^2 * \vec{\mathcal{B}}, \ldots, \mathcal{G}^{k-1} * \vec{\mathcal{B}}].$$

We assume that $k \leq \min(\ell, m)$, that the tubal rank of each singular tuple of $\mathcal{G}$ is $\min(\ell, m)$, and that when $\vec{\mathcal{B}}$ is normalized, the normalization term is an invertible tubal scalar. We consider here only the case where (using the definition in the previous section) $\dim(N(\mathcal{K}_k)) = 0$, which should ensure that we have no "breakdowns" (see below for definition).

The basic idea is to directly extend Krylov iterations by replacing matrix-vector products by t-products between the tensor and a matrix. Based on the discussion in the previous subsection, this approach is equivalent to applying a matrix Krylov method on each of the blocks $\hat{G}^{(i)}$ in the block diagonal matrix (in Fourier space) simultaneously, and therefore to get the method to work the way we want, we have to normalize appropriately.

Suppose $\mathcal{A}$ is symmetric positive definite. Then the Symmetric Lanczos iteration (compare to the matrix version [31, Algorithm 36.1]) is as shown in Algorithm 5, which implicitly generates a set of orthogonal generating matrices for $R(\mathcal{K}_k(\mathcal{A}, \mathcal{B}))$.

Since we have assumed for simplicity that the algorithm does not break down (i.e., that $\mathbf{c}_i$ is always invertible), in exact arithmetic taken to k steps this would

Algorithm 5: Symmetric Lanczos iteration.

$\vec{\mathcal{Q}}_0 \leftarrow 0$.
$[\vec{\mathcal{Q}}_1, \mathbf{z}_0] \leftarrow \text{Normalize}(\vec{\mathcal{B}})$
for $i = 1$ **to** ...
  $\vec{\mathcal{V}} \leftarrow \mathcal{A} * \vec{\mathcal{Q}}_i$
  $\mathbf{c}_i \leftarrow \vec{\mathcal{Q}}_i^T * \vec{\mathcal{V}}$
  $\vec{\mathcal{V}} \leftarrow \vec{\mathcal{V}} - \vec{\mathcal{Q}}_{i-1} * \mathbf{z}_{i-1} - \vec{\mathcal{Q}}_i * \mathbf{c}_i$
  $[\vec{\mathcal{Q}}_{i+1}, \mathbf{z}_i] \leftarrow \text{Normalize}(\vec{\mathcal{V}})$
end for

produce $\mathcal{Q}_k^T * \mathcal{A} * \mathcal{Q}_k = \mathcal{T}$, where

$$\mathcal{T} = \begin{bmatrix} \mathbf{c}_1 & \mathbf{z}_1 & 0 & 0 & 0 \\ \mathbf{z}_1 & \mathbf{c}_2 & \mathbf{z}_2 & \cdots & 0 \\ 0 & \mathbf{z}_2 & \mathbf{c}_3 & \ddots & \\ \vdots & \ddots & \ddots & \ddots & \mathbf{z}_{k-1} \\ 0 & \cdots & 0 & \mathbf{z}_{k-1} & \mathbf{c}_k \end{bmatrix}$$

is a tridiagonal $k \times k \times n$ tensor. Suppose that $\mathcal{T} = \mathcal{V} * \mathcal{D} * \mathcal{V}^T$; then in analogy with matrix computation, we call lateral slices of $\mathcal{Q}_k * \mathcal{V}$ the Ritz matrices. Note that when k is small, we can possibly afford to find the eigendecomposition of $\mathcal{T}$ directly: we need only compute $O(k)$ FFTs and inverse FFTs of length n, then $O(n/2)$ tridiagonal matrix eigenvalue problems (in the Fourier domain).

Golub–Kahan iterative bidiagonalization (often referred to as Lanczos bidiagonalization), which implicitly generates an orthogonal matrix generating set for $\mathcal{K}_k(\mathcal{A}^T * \mathcal{A}, \mathcal{A}^T * \vec{\mathcal{B}})$, can likewise be extended to the tensor case (see Algorithm 6).

Algorithm 6: Golub–Kahan iterative bidiagonalization.

Input: $\mathcal{A} \in \mathbb{K}_n^{\ell \times m}$, $\vec{\mathcal{B}} \in \mathbb{K}_n^m \neq 0$

$\vec{\mathcal{Q}}_0 \leftarrow 0$.
$[\vec{\mathcal{Q}}_1, \mathbf{z}_1] \leftarrow \text{Normalize}(\vec{\mathcal{B}})$ with $\mathbf{z}_1$ invertible
for $i = 1$ **to** ...
  $\vec{\mathcal{W}}_i \leftarrow \mathcal{A}^T * \vec{\mathcal{Q}}_i - \vec{\mathcal{W}}_{i-1} * \mathbf{z}_i$
  $[\vec{\mathcal{W}}_i, \mathbf{c}_i] \leftarrow \text{Normalize}(\vec{\mathcal{W}}_i)$
  $\vec{\mathcal{Q}}_{i+1} \leftarrow \mathcal{A} * \vec{\mathcal{W}}_i - \vec{\mathcal{Q}}_i * \mathbf{c}_i$
  $[\vec{\mathcal{Q}}_{i+1}, \mathbf{z}_{i+1}] \leftarrow \text{Normalize}(\vec{\mathcal{Q}}_{i+1})$
end for

The algorithm after k steps and no breakdowns produces the decomposition

$$\mathcal{A} * \mathcal{W} = \mathcal{Q} * \mathcal{P},$$

where $\mathcal{W}$ and $\mathcal{Q}$ have k and $k+1$ orthogonal lateral slices, respectively, and $\mathcal{P}$ is a $(k+1) \times k \times n$ tensor. Similar to the case above, we get approximate SVDs from the Ritz triples that are formed using the t-SVDs of $\mathcal{P}$. Thus if $\mathcal{P} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$, the approximate singular tuples and the corresponding right and left singular matrices are given by

$$(\mathbf{s}_i, \mathcal{W} * \vec{\mathcal{V}}_i, \mathcal{Q} * \vec{\mathcal{U}}_i)$$

for i up to k. Since $\mathcal{P}$ is facewise bidiagonal, $\hat{\mathcal{P}}$ is facewise bidiagonal, and so computing the t-SVD of $\mathcal{P}$ from scratch amounts to computing $O(k)$ forward and inverse FFTs, and $O(n/2)$ SVDs of bidiagonal matrices.[5]

[5]Since $\hat{\mathcal{P}}$ does not change values in its leading principle submatrices, bidiagonal matrix SVD updating techniques could be employed to make this process more efficient from one iteration to the next.

The final question we are interested in is in developing Krylov-iterative solvers. In other words, we would like to consider solutions to problems of the form

$$\mathcal{A} * \vec{\mathcal{X}} = \vec{\mathcal{B}} \text{ or } \min_{\vec{\mathcal{X}}} \|\mathcal{A} * \vec{\mathcal{X}} - \vec{\mathcal{B}}\|_F,$$

where we restrict our approximate solution so that it is a t-linear combination of the columns of a Krylov tensor, $\vec{\mathcal{X}} = \mathcal{K}_k * \vec{\mathcal{C}}$, and $\mathcal{K}_k$ is generated by $\mathcal{A}, \vec{\mathcal{B}}$ (if $\mathcal{A}$ is symmetric positive definite) or $\mathcal{A}^T * \mathcal{A}, \mathcal{A}^T * \vec{\mathcal{B}}$, respectively.

Working backwards, if we can run a conjugate gradient (CG) iteration on each block in the Fourier domain, we ought to be able to specify a CG iteration for tensors. An alternative view would be to develop a CG algorithm directly from the symmetric Lanczos iteration. We present one implementation of the CG algorithm here (see Algorithm 7; compare to [31, Algorithm 38.1], for example). The one difference with tensor CG vs. matrix CG is that the expression $\vec{\mathcal{X}}^T * \mathcal{A} * \vec{\mathcal{X}}$ is a tubal scalar rather than a scalar (see the discussion of bilinear forms in section 3). If $\mathcal{A}$ is symmetric positive definite, then since the tensor CG routine is in exact arithmetic doing CG on n, $m \times m$ subproblems in the Fourier domain, the $\hat{A}^{(i)}$-norm of the error is being reduced on each of the subproblems as a function of k. Thus, the Fourier coefficients of $\vec{\mathcal{X}}_k^T * \mathcal{A} * \vec{\mathcal{X}}_k$ are being reduced as a function of k, from which it follows that $\|\vec{\mathcal{X}}_k^T * \mathcal{A} * \vec{\mathcal{X}}_k\|$ is decreasing. In exact arithmetic, using our tensor definition of orthogonality, we do still get orthogonality of the residual matrices, and $\mathcal{A}$-conjugacy of the search directions. Note that we perform a normalization step initially to prevent growth in factors.

Algorithm 7: t-CG.

Input: $\mathcal{A} \in \mathbb{K}_n^{m \times m}$ positive definite, $\vec{\mathcal{B}} \in \mathbb{K}_n^m \neq 0$
Output: Estimate $\vec{\mathcal{X}}$, $\mathcal{A}$-conjugate $\vec{\mathcal{P}}_i$, orthogonal $\vec{\mathcal{R}}_i$

$\vec{\mathcal{X}}_0 \leftarrow 0$.
$[\vec{\mathcal{R}}_0, \mathbf{a}] \leftarrow \text{Normalize}(\vec{\mathcal{B}})$; $\vec{\mathcal{P}}_0 \leftarrow \vec{\mathcal{R}}_0$.
for $i = 1$ **to** ...
  $\mathbf{c} \leftarrow (\vec{\mathcal{P}}^T * \mathcal{A} * \vec{\mathcal{P}})^{-1} * (\vec{\mathcal{R}}^T * \vec{\mathcal{R}})$
  $\vec{\mathcal{X}}_i \leftarrow \vec{\mathcal{X}}_{i-1} + \vec{\mathcal{P}}_{i-1} * \mathbf{c}$
  $\vec{\mathcal{R}}_i \leftarrow \vec{\mathcal{R}}_{i-1} - \mathcal{A} * (\vec{\mathcal{P}}_i * \mathbf{c})$
  $\mathbf{d} \leftarrow (\vec{\mathcal{R}}_{i-1}^T * \vec{\mathcal{R}}_{i-1})^{-1} * (\vec{\mathcal{R}}_i^T * \vec{\mathcal{R}}_i)$
  $\vec{\mathcal{P}}_i \leftarrow \vec{\mathcal{R}}_i + \vec{\mathcal{P}}_{i-1} * \mathbf{d}$
end for
$\vec{\mathcal{X}} \leftarrow \vec{\mathcal{X}} * \mathbf{a}$.

7. Numerical examples. In this section we give two examples illustrating the potential of the previously developed theoretical and computational framework.

7.1. Image deblurring. The most basic model of linear image blurring is given via the matrix vector equation

$$Kx + e = b,$$

where $x := \texttt{vec}(X)$ is the vectorized version of the original image, K represents the known blurring operator, and $b := \texttt{vec}(B)$ is the vectorized version of the recorded

blurry and noisy image. The noise vector e is usually assumed to be white and unknown. In many deblurring applications, the matrix K has significant structure. For purposes of illustrating our Krylov solver algorithm, we will assume that K is block circulant, which is not an uncommon assumption and is consistent with a model for spatially invariant blur given periodic boundary conditions in that direction.

The presence of the noise forces one to use regularization to damp the effects of the noise and make the system better conditioned. The most well-known type of regularization is Tikhonov regularization, in which case one solves

$$\min_x \|Kx - b\|_2^2 + \lambda^2 \|x\|_2^2,$$

and λ controls the amount of damping.

The normal equations for this optimization problem are

$$(K^T K + \lambda^2 I)x = K^T b.$$

The assumed structure on K ensures that $K^T K + \lambda^2 I$ is still a block circulant matrix. Given this structure, as discussed in [16], an equivalent way to formulate the problem above is

$$(\mathcal{A}^T * \mathcal{A} + \mathcal{D}[\mathbf{d}]) * \vec{\mathcal{X}} \approx \mathcal{A}^T * \vec{\mathcal{B}},$$

where $\mathcal{A}$ is obtained by stacking the first block column of K, and $\vec{\mathcal{X}} = \texttt{twist}(X)$, $\vec{\mathcal{B}} = \texttt{twist}(B)$, $\mathbf{d} = \lambda^2 \mathbf{e}_1$, and $\mathcal{D}[\mathbf{d}]$ is the diagonal tensor with $\mathbf{d}$ repeated down the diagonal. For more on exploiting circulant structure with respect to tensor computations in image processing, see [26].

Since the purpose of this example is to illustrate the performance of the tensor-CG algorithm, we will assume that a reasonable value of λ is already known—choosing regularization parameters is an important and well-researched topic and far beyond the scope of this paper. The parameter we chose in this problem was simply selected by a very few trial and error experiments.

In our test problem, we will consider a 128 by 128 satellite image. We construct a blurring matrix K that is block circulant with Toeplitz blocks as described in [16] (specifically, K here is equivalent to $\tilde{A}$ in that paper) and fold the first block column to obtain $\mathcal{A}$. The true image X is given in left of Figure 7.1. The blurred, noisy image is created first by computing $\mathcal{A} * \vec{\mathcal{X}}$, where $\vec{\mathcal{X}} = \texttt{twist}(X)$, then adding random Gaussian noise with a noise level of 0.1 percent. The blurred noisy image is shown in the middle of Figure 7.1. We set $\lambda^2 = 2$ and find that after 42 iterations, the minimum relative error between $\vec{\mathcal{X}}$ and the image estimate is obtained. The resulting reconstruction is given in the rightmost subplot of Figure 7.1.

In this problem, since we did not have the exact solution to the normal equations, we could not track $\|\vec{\mathcal{E}}^T * (\mathcal{A}^T * \mathcal{A} + \mathcal{D}[\mathbf{d}]) * \vec{\mathcal{E}}\|$, where $\vec{\mathcal{E}}$ is the error between the exact and computed solutions. However, assuming λ^2 is a reasonable value, $\vec{\mathcal{E}} \approx \vec{\mathcal{X}} - \vec{\mathcal{X}}_k$, where $\vec{\mathcal{X}}_k$ denotes the kth iterate, and plugging this estimate in for $\vec{\mathcal{E}}$, the resulting approximation to $\|\vec{\mathcal{E}}^T * (\mathcal{A}^T * \mathcal{A} + \mathcal{D}[\mathbf{d}]) * \vec{\mathcal{E}}\|$ is given in Figure 7.2. It is indeed decreasing, and stagnates where we appear to have reached the optimal regularized solution.

7.2. Projections and dimensionality reduction. In this experiment, we are provided with a set of 128 gray scale training images of size 128×128. The images are of 3 different objects being rotated through a single degree of freedom throughout

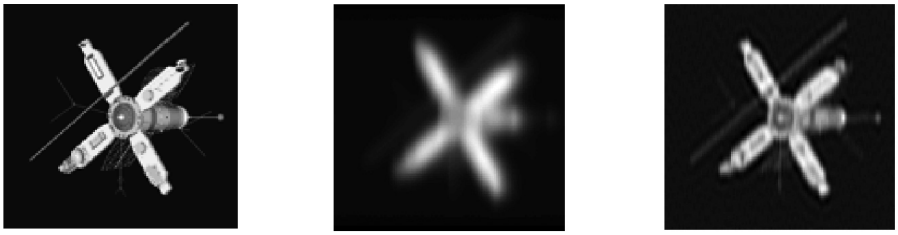

FIG. 7.1. *True image (left), blurred noisy image (middle), reconstruction after* 42 *iterations our tensor CG algorithm on the regularized normal equations with* $\lambda^2 = 2$ *(right).*

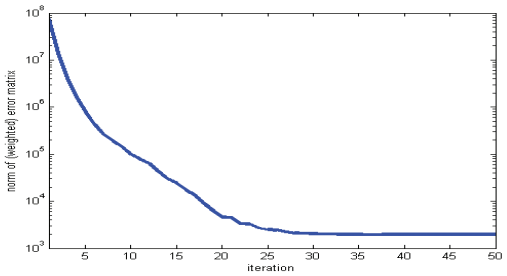

FIG. 7.2. *Estimate of the norm of the error as a function of iteration.*

360° and were generated by ray-tracing high fidelity CAD models provided by [21]. We construct a normalized database of these images by first subtracting the mean of all the images from each individual image and then concatenating them as lateral slices into the tensor $\mathcal{A}$, i.e., $\mathcal{A} = [\vec{\mathcal{Y}}_1, \vec{\mathcal{Y}}_2, \ldots, \vec{\mathcal{Y}}_n]$, where $\vec{\mathcal{Y}}_i$ represents the ith image tensor.

In a real object classification (recognition and pose estimation) scenario, when a new (normalized) image $\vec{\mathcal{Y}}$ becomes available, we are interested in recognizing the object within the image as well as determining its orientation. The naive approach to this problem would be to compare the new image $\vec{\mathcal{Y}}$ with each image in $\mathcal{A}$ to classify the object. Unfortunately, due to the shear size of the image database, this approach is computationally infeasible. Thus, the desire is to compress the database, and one way to do this would be to truncate the t-SVD of $\mathcal{A}$ to provide the best rank-k approximation [2, 10, 11]. The implication is that the first k lateral slices of $\mathcal{U}$ contain enough information about $\mathcal{A}$ that the object of interest can be reconstructed by computing the coefficients of the projection onto the range of those lateral slices. In other words, because $\vec{\mathcal{A}}_j \approx \mathcal{U}_k(\vec{\mathcal{C}}_j)$, where $\vec{\mathcal{C}}_j = \mathcal{U}_k^T * \vec{\mathcal{A}}_j$ and $\mathcal{U}_k$ is the tensor containing the first k lateral slices of $\mathcal{U}$, object classification can be carried out by comparing $\vec{\mathcal{C}}_j$ with $\mathcal{U}_k^T * \vec{\mathcal{Y}}$. Furthermore, if the object is in the database and k is sufficiently large, then $\vec{\mathcal{Y}} \approx \left(\mathcal{U}_k * \mathcal{U}_k^T\right) * \vec{\mathcal{Y}}$ provides a rank-k reconstruction of $\vec{\mathcal{Y}}$ once the mean image is added to $\vec{\mathcal{Y}}$. This reconstruction can serve as an estimate to what dimension k is required for accurate classification. Note that the tensor-tensor product $\mathcal{U}_k * \mathcal{U}_k^T$ is an orthogonal projector as defined in section 5.

An alternate method for determining an appropriate dimension k required for accurate object classification was defined for the matrix SVD in [5] and is referred

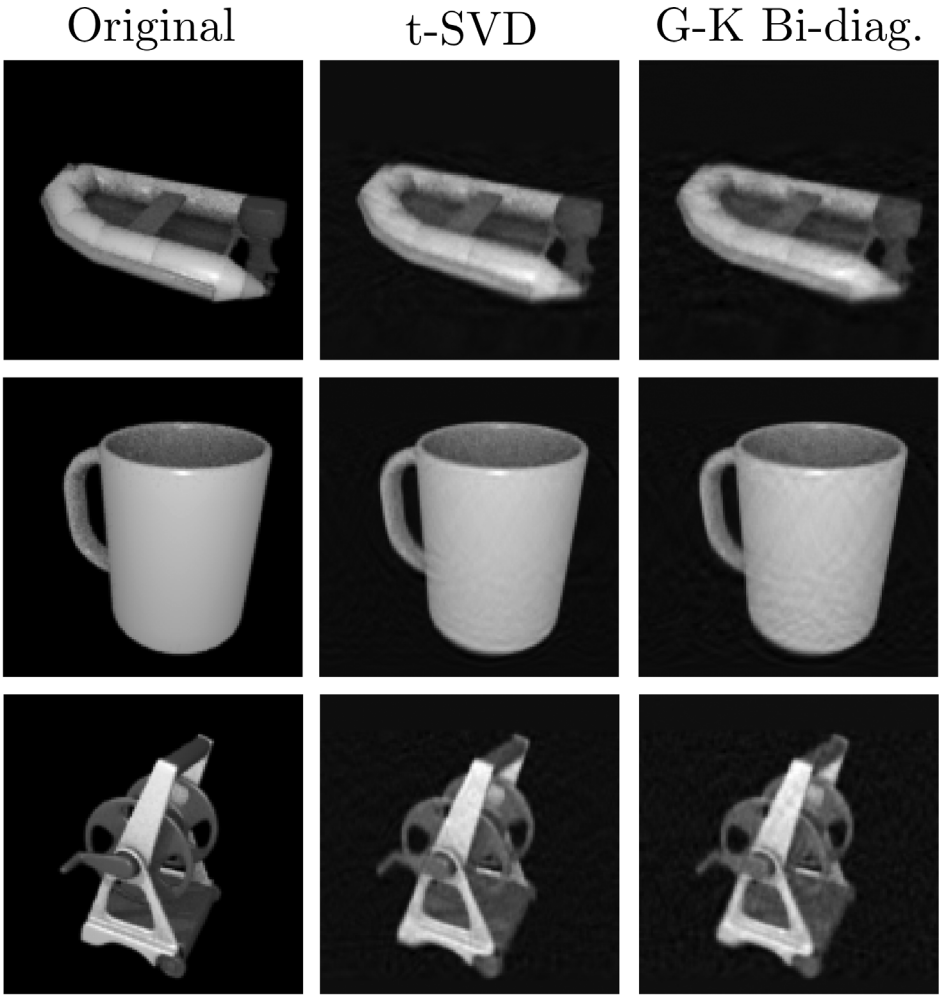

FIG. 7.3. *Reconstructions of certain poses of objects in the image database $\mathcal{A}$. The leftmost column shows the images to be reconstructed, the middle column shows images reconstructed using the t-SVD, and the rightmost column shows images the reconstructed using the G-K bidiagonalization. As is apparent from the figure, for $k = 20$, both the t-SVD as well as the $\mathcal{U}_k$ computed using the G-K bidiagonalization are capable of reconstructing the original images quite well.*

to as the energy recovery ratio. The energy recovery ratio was extended to tensors in [14] and is computed as

$$\rho(\mathcal{A},\mathcal{U}_k) = \frac{\sum_{i=1}^{k} \|\mathbf{s}_i\|_F^2}{\|\mathcal{A}\|_F^2}, \tag{7.1}$$

where $\mathbf{s}_i$ is the ith singular tuple. Note that because the lateral slices of $\mathcal{U}$ are orthonormal, the energy recovery ratio $\rho \leq 1$ and achieves a maximum value when $k = n$.

To illustrate the tensor Golub–Kahan (G-K) iterative bidiagonalization algorithm described in section 6, a random input image $\vec{\mathcal{Y}}$ is chosen for each of the three objects in the database. A rank $k = 20$ reconstruction is then performed using the capability

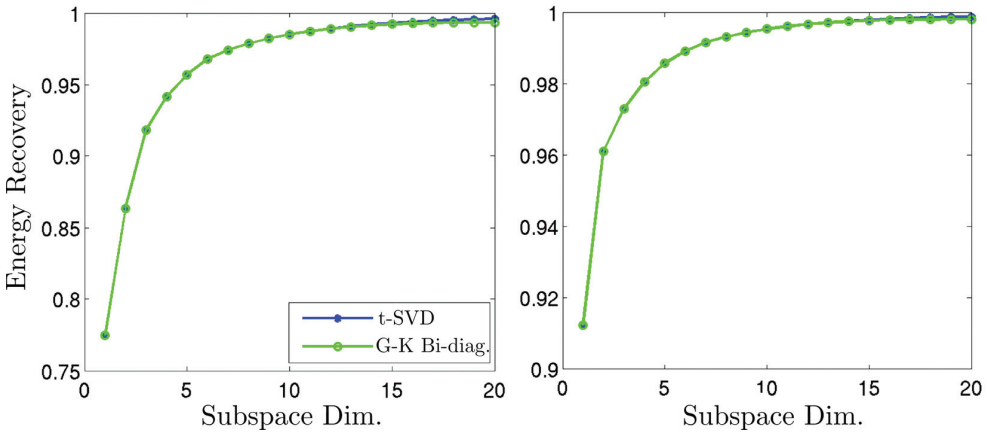

FIG. 7.4. *Energy recovery plots for the objects shown in Figure* 7.3 (left: boat; right: cup) *comparing the energy recovered using the estimated left-singular matrices* $\mathcal{U}_k$ *computed via the G-K bidiagonalization and the true left-singular matrices computed via the t-SVD. Note that the energy recovered using the estimates is nearly identical to that computed using the t-SVD. A nearly identical pattern is observed for the hose reel image.*

of the left-singular matrices $\mathcal{U}_k$ as computed by the t-SVD as well as their estimates as computed by the G-K bidiagonalization. The results of this experiment are illustrated in Figure 7.3, where the leftmost images, the images are to be reconstructed, the middle images illustrate the reconstruction using the t-SVD, and the rightmost images illustrate the reconstruction using the G-K bidiagonalization. As is apparent from the figure, for $k = 20$, both the t-SVD as well as the $\mathcal{U}_k$ computed using the G-K bidiagonalization are capable of reconstructing the original image.

To compare the accuracy of the estimates of $\mathcal{U}_k$ computed using the G-K bidiagonalization to those computed using the t-SVD, the energy recovery ratio was used. Figure 7.4 shows the results of this comparison for a dimension $k = 20$. As can be seen from the figure, the energy recovered using the G-K generated estimates is almost exactly the same as that recovered using the t-SVD. Notice that divergence of the two energy recovery plots occurs only near the last 4–5 dimensions (i.e., dimensions 15–20). For analysis of this phenomenon for larger k, we need to consider that convergence is not happening on each face of $\hat{A}^{(i)}$ at the same rate. The study of convergence is left for future work.

8. Conclusions. The purpose of this paper was twofold. First, we set up the necessary theoretical framework for tensor computation by delving further into the insight explored in [1] in which the author considered tensors as operators on a set of matrices. As our numerical examples illustrate, there are applications that benefit from treating inherently two-dimensional objects (e.g., images) in their multidimensional format. Thus, our second purpose was to demonstrate how one might build algorithms around our new concepts and constructs to tackle practical problems. To this end, we were able to extend familiar matrix algorithms (e.g., power, Krylov subspace iteration) to tensors. Clearly, more work needs to be done to analyze the behavior of such algorithms in finite precision arithmetic and to enhance their performance—for instance, adding multiple shifts to power iteration. Also, monitoring the estimates of the condition vector introduced here may help to explain when the problem could be "projected" down to a problem of smaller dimension, as certain components may have

already converged. We anticipate that further study will reveal additional applications where the framework presented here will indeed prove valuable.

Appendix A. A new matrix norm. A nice consequence of the length definition given in Definition 3.2 is that it in fact *defines a valid matrix norm*[6] on X. To prove this fact, we first need the following lemma.

LEMMA A.1. *For* $\vec{\mathcal{U}} := \texttt{twist}(U), U \in \mathbb{R}^{m\times n}$, *we have*

$$\begin{aligned} \|\langle \vec{\mathcal{U}}, \vec{\mathcal{U}}\rangle\|_F &= \|\vec{\mathcal{U}}^T * \vec{\mathcal{U}}\|_F \\ &= \|\texttt{bcirc}(\vec{\mathcal{U}}^T)\texttt{unfold}(\vec{\mathcal{U}})\|_2 \\ &\le \|\texttt{bcirc}(\vec{\mathcal{U}}^T)\|_2\|u\|_2 \\ &\le \|\texttt{bcirc}(\vec{\mathcal{U}})\|_2\|u\|_2, \end{aligned} \tag{A.1}$$

where $u = \texttt{unfold}(\vec{\mathcal{U}}) \in \mathbb{R}^{mn}$, *which implies*

$$\frac{\|\langle \vec{\mathcal{U}}, \vec{\mathcal{U}}\rangle\|_F}{\|u\|_2} \le \|\texttt{bcirc}(\vec{\mathcal{U}})\|_2. \tag{A.2}$$

THEOREM A.2. *Let* $X \in \mathbb{R}^{m\times n}$. *Define* $\|X\|_d = \|\vec{\mathcal{X}}\|$, *where the right-hand side is as defined above. Then* $\|X\|_d$ *is a valid matrix norm on* $\mathbb{R}^{m\times n}$.

Proof. The fact that $\|X\|_d \ge 0$ and $\|X\|_d = 0$ iff $X = 0$ follows from the definition. Also, if $c \in \mathbb{R}$, $cX \equiv \vec{\mathcal{X}} * \mathbf{c}$, where $\mathbf{c}$ has c on the first face and zeros on the other faces. Thus, it's easy to see that the denominator $\|\vec{\mathcal{X}} * c\|_F = |c|\|\vec{\mathcal{X}}\|_F$. In the numerator, $\langle \vec{\mathcal{X}} * c, \vec{\mathcal{X}} * c\rangle = (\mathbf{c}^T * \mathbf{c}) * \langle \vec{\mathcal{X}}, \vec{\mathcal{X}}\rangle$, which means that every element of the tubal scalar $\langle \vec{\mathcal{X}}, \vec{\mathcal{X}}\rangle$ is multiplied by c^2, and so $\|\langle \vec{\mathcal{X}} * \mathbf{c}, \vec{\mathcal{X}} * \mathbf{c}\rangle\|_F = c^2\|\langle \vec{\mathcal{X}}, \vec{\mathcal{X}}\rangle\|_F$. It follows that $\|cX\|_d = |c|\|X\|_d$.

The real work is in establishing the triangle inequality. Observe that

$$\|X + Y\|_d = \frac{\langle \vec{\mathcal{X}} + \vec{\mathcal{Y}}, \vec{\mathcal{X}} + \vec{\mathcal{Y}}\rangle\|_F}{\|x + y\|_2} \le \|\texttt{bcirc}(\vec{\mathcal{X}} + \vec{\mathcal{Y}})\|_2,$$

where the latter inequality follows from (A.2) in the lemma and $x = \texttt{unfold}(\vec{\mathcal{X}}), y = \texttt{unfold}(\vec{\mathcal{Y}})$. But

$$\|\texttt{bcirc}(\vec{\mathcal{X}} + \vec{\mathcal{Y}})\|_2 \le \|\texttt{bcirc}(\vec{\mathcal{X}})\|_2 + \|\texttt{bcirc}(\vec{\mathcal{Y}})\|_2$$

by the triangle inequality on the matrix 2-norm. However,

$$\|\texttt{bcirc}(\vec{\mathcal{X}})\|_2 + \|\texttt{bcirc}(\vec{\mathcal{Y}})\|_2 \le \frac{\|\texttt{bcirc}(\vec{\mathcal{X}})x\|_2}{\|x\|_2} + \frac{\|\texttt{bcirc}(\vec{\mathcal{Y}})y\|_2}{\|y\|_2}$$

follows from the definition of the matrix 2-norm as an induced matrix norm. Since the first term on the right-hand side above, by the lemma, is $\|\vec{\mathcal{X}}\|$ and the second is $\|\vec{\mathcal{Y}}\|$, the proof is complete. ∎

Appendix B. Discussion of angles between elements of $\mathbb{K}^{\boldsymbol{m}}_{\boldsymbol{n}}$. In the next two examples, we show why it is important to keep track of an entire tuple of angles.

Example B.1. Let $X = \left[\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right]$, $Y = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right]$. Then

$$\cos(\boldsymbol{\theta})^{(1)} = 0, \qquad \cos(\boldsymbol{\theta})^{(2)} = 1.$$

[6] Thanks to Christino Tamon for suggesting we try to prove this.

Example B.2. *Let* $X = \left[\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right]$, $Y = \left[\begin{smallmatrix} 0 & 0 \\ 1 & 0 \end{smallmatrix}\right]$. *Then*

$$\cos(\boldsymbol{\theta})^{(1)} = 0, \qquad \cos(\boldsymbol{\theta})^{(2)} = 0.$$

If we let $\mathcal{A}(:,1,:) = X; \mathcal{A}(:,2,:) = Y$, then in the first case, the tensor $\mathcal{A}$ is not invertible (and thus certainly not orthogonal), while in the second, the tensor $\mathcal{A}$ so formed is orthogonal. This might seem odd, since in both examples $\texttt{vec}(X) \perp \texttt{vec}(Y)$. The difference is that, in vectorizing, we would remove the inherent multidimensional nature of the objects. As we will see, it is possible to construct any 2×2 matrix from an appropriate combination of the X and Y in the second example, but it is not be possible to construct any 2×2 matrix from the same type of combination using X and Y in the first example.

The entries in the tubal angle are a measure of how close to the Z^{j-1}-conjugate each of the rows of X, Y are to each other. Using the definition of $*$, $X = \texttt{squeeze}(\vec{\mathcal{X}})$, and using x_j to denote a column of X (with similar definitions for Y and y_j),

$$\langle \vec{\mathcal{X}}, \vec{\mathcal{Y}} \rangle^{(j)} = \sum_{i=1}^{m} x_i^T Z^{j-1} y_i = \sum_{i=1}^{m} (x_i^T F_n^H)(F_n Z^{j-1} F_n^H)(F_n y_i) = \sum_{i=1}^{m} \texttt{conj}(\hat{x}_i)^T \hat{Z}^{j-1} \hat{y}_i,$$

where Z is the upshift matrix, and $\hat{Z}^{j-1} = F_n Z^{j-1} F_n^H$ is a diagonal matrix with powers of roots of unity on the diagonal, since Z is circulant.

In particular, if X, Y are such that their respective rows are all pairwise orthogonal to each other, $\langle \vec{\mathcal{X}}, \vec{\mathcal{Y}} \rangle^{(1)} = 0 = \langle \vec{\mathcal{Y}}, \vec{\mathcal{X}} \rangle^{(1)}$, as in these two examples. However, in the first example, $x_1^T Z y_1 = 1; x_2^T Z y_2 = 0$, whereas $x_1^T Z y_1 = 0 = x_2^T Z y_2$ in the second example.

Thus, to truly have a consistency in the definition of a pair of orthogonal elements of $\mathbb{K}_n^m$, we need all the angles to be $\pi/2$. That is, an orthogonal set in $\mathbb{K}_n^m$ should be characterized by the fact that they have the highest degree of rowwise conjugacies possible, which implicitly accounts for our being able to describe any element of $\mathbb{K}_n^m$ with only m elements using the notion of a t-linear combination.

Note the following:

- For each element of the orthonormal set, $\|\vec{\mathcal{U}}_i\| = 1$.
- If $\vec{\mathcal{U}}_i$ and $\vec{\mathcal{U}}_j$ are orthogonal, $\boldsymbol{\theta}^{(j)} = \pi/2, j = 1, \ldots, n$.
- In general, some of the angles will be nonzero, even if $\vec{\mathcal{X}} = \vec{\mathcal{Y}}$, which is in contrast to standard linear algebra, when we expect that if $v \in \mathbb{R}^n$ is a nonzero vector, the angle between v and itself is zero. However, if $\vec{\mathcal{X}} = \vec{\mathcal{Y}}$, *the first entry in the tubal angle will be zero.* This is due to the fact that the first component of the tubal angle is the angle between $\texttt{vec}(X)$ and $\texttt{vec}(Y)$. In particular, if $\langle \vec{\mathcal{X}}, \vec{\mathcal{X}} \rangle = \alpha \mathbf{e}_1$, the first angle is zero, while the rest are $\pi/2$.

REFERENCES

[1] K. Braman, *Third-order tensors as linear operators on a space of matrices*, Linear Algebra Appl., 433 (2010), pp. 1241–1253.

[2] K. Braman and R. Hoover, *Tensor decomposition and application in signal and image processing*, abstract, Invited Minisymposium Presentation, 2010 SIAM Annual Meeting; http://www.siam.org/meetings/an10/An10abstracts.pdf, p. 58, MS35.

[3] K. Braman and M. Kilmer, *Householder abstract booklet*, abstract and talk, Householder Symposium, 2011.

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

[5] C.-Y. Chang, A. A. Maciejewski, and V. Balakrishnan, *Fast eigenspace decomposition of correlated images*, IEEE Trans. Image Process., 9 (2000), pp. 1937–1949.
[6] P. Comon, *Tensor decompositions*, in Mathematics in Signal Processing V, J. G. McWhirter and I. K. Proudler, eds., Clarendon Press, Oxford, UK, 2002, pp. 1–24.
[7] L. De Lathauwer and B. De Moor, *From matrix to tensor: Multilinear algebra and signal processing*, in Mathematics in Signal Processing IV, J. McWhirter and I. K. Proudler, eds., Clarendon Press, Oxford, UK, 1998, pp. 1–15.
[8] L. De Lathauwer, B. De Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278.
[9] D. F. Gleich, C. Greif, and J. M. Varah, *The power and Arnoldi methods in an algebra of circulants*, Numer. Linear Algebra Appl., doi: 10.1002/nla.1845.
[10] N. Hao and M. Kilmer, *Tensor-SVD with applications in image processing*, Invited Minisymposium Presentation, 2010 SIAM Annual Meeting; http://www.siam.org/meetings/an10/An10abstracts.pdf, p. 58, MS35.
[11] N. Hao, M. E. Kilmer, K. Braman, and R. C. Hoover, *Facial recognition using tensor-tensor decompositions*, SIAM J. Imaging Sci., 6 (2013), pp. 437–463.
[12] 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.
[13] W. S. Hoge and C.-F. Westin, *Identification of translational displacements between N-dimensional data sets using the high order SVD and phase correlation*, IEEE Trans. Image Process., 14 (2005), pp. 884–889.
[14] R. C. Hoover, K. S. Braman, and N. Hao, *Pose estimation from a single image using tensor decomposition and an algebra of circulants*, in Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), 2011, pp. 2928–2939.
[15] M. E. Kilmer, K. Braman, and N. Hao, *Third Order Tensors as Operators on Matrices: A Theoretical and Computational Framework with Applications in Imaging*, Tech. Report CS-TR-01-11, Tufts University, 2011.
[16] M. E. Kilmer and C. D. Martin, *Factorization strategies for third-order tensors*, Linear Algebra Appl., 435 (2011), pp. 641–658.
[17] M. E. Kilmer, C. D. Martin, and L. Perrone, *A Third-Order Generalization of the Matrix SVD as a Product of Third-Order Tensors*, Tech. Report TR-2008-4, Tufts University, 2008.
[18] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.
[19] T. G. Kolda and J. R. Mayo, *Shifted power method for computing tensor eigenpairs*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1095–1124.
[20] P. Kroonenberg, *Three-Mode Principal Component Analysis: Theory and Applications*, DSWO Press, Leiden, 1983.
[21] K. Legaz, 3-*D Model Database for Blender*, http://3dmodels.katorlegaz.com/, 2007.
[22] L.-H. Lim, *Singular values and eigenvalues of tensors: A variational approach*, in CAMSA P'05: Proceeding of the IEEE International Workshop on Computational Advances in Multi-sensor Adaptive Processing, 2005, pp. 129–132.
[23] J. Nagy and M. Kilmer, *Kronecker product approximation for preconditioning in three-dimensional imaging applications*, IEEE Trans. Image Process., 15 (2006), pp. 604–613.
[24] M. Ng, L. Qi, and G. Zhou, *Finding the largest eigenvalue of a nonnegative tensor*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1090–1099.
[25] L. Qi, *Eigenvalues and invariants of tensors*, J. Math. Anal. Appl., 325 (2007), pp. 1363–1367.
[26] M. Rezghi and L. Eldèn, *Diagonalization of tensors with circulant structure*, Linear Algebra Appl., 435 (2011), pp. 422–447
[27] B. Savas and L. Eldén, *Krylov Subspace Methods for Tensor Computations*, Tech. Report LITH-MAT-R-2009-02-SE, Department of Mathematics, Linköpings Universitet, 2009.
[28] B. Savas and L. Eldén, *Krylov-type methods for tensor computations* I, Linear Algebra Appl., 438 (2013), pp. 891–918.
[29] N. Sidiropoulos, R. Bro, and G. Giannakis, *Parallel factor analysis in sensor array processing*, IEEE Trans. Signal Process., 48 (2000), pp. 2377–2388.
[30] A. Smilde, R. Bro, and P. Geladi, *Multi-way Analysis: Applications in the Chemical Sciences*, John Wiley, Chichester, UK, 2004.
[31] L. N. Trefethen and D. Bau, III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
[32] L. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.
[33] M. Vasilescu and D. Terzopoulos, *Multilinear analysis of image ensembles: Tensorfaces*, in

Proceedings of the 7th European Conference on Computer Vision (ECCV 2002), Lecture Notes in Comput. Sci. 2350, Springer, New York, 2002, pp. 447–460.

[34] M. Vasilescu and D. Terzopoulos, *Multilinear image analysis for facial recognition*, in Proceedings of the 16th International Conference on Pattern Recognition (ICPR 2002), Vol. 2, IEEE Computer Society Press, 2002, pp. 511–514.

[35] M. Vasilescu and 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), IEEE Computer Society Press, 2003, pp. 93–99.