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

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

Abstract.
Recent work by Kilmer and Martin, [10] and Braman [2] 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 [2]. 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 facial recognition and solving a regularized image deblurring problem.

Key words. eigendecomposition, tensor decomposition, singular value decomposition, multidimensional arrays

AMS subject classifications. 15A69, 65F30

1. Introduction. The term *tensor*, as used in the context of this paper, refers to a multi-dimensional 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 [24], psychometrics [13], and image and signal processing [5, 15, 23, 17, 8, 20, 29, 28, 30]. Various tensor decompositions such as CANDECOMP/PARAFAC (CP) [4, 7], TUCKER [26], and Higher-Order SVD [14] 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 [11].

Recent work by Kilmer, Martin [10] and Braman [2] provides an alternative setting in which the familiar tools of linear algebra can be extended to better understand third-order tensors. Specifically, in [9] and [10] the authors define a multiplication operation which is closed on the set of third-order tensors. This multiplication allows tensor factorizations which are analogs of matrix factorizations such as SVD, QR and eigendecompostions. In addition, [2] 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

*This work was supported by NSF grant 0914957 .

†Department of Mathematics, Tufts University, 113 Bromfield-Pearson Bldg., Medford, MA 02155, misha.kilmer@tufts.edu,

‡Department of Mathematics and Computer Science, South Dakota School of Mines and Technology, MacLaury 203B, karen.braman@sdsmt.edu,

§Department of Mathematics, Tufts University, Medford, MA 02155, ning.hao@tufts.edu

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 should note that we are not the first to consider extensions of Krylov methods to tensor computation. Other recent work in this area includes the work in [21, 22]. Our methods are different than their work, however, because we rely on the t-product construct in [10] 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 are consistent with our new theoretical framework. The numerical results are intended to show the potential suitablity 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.

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, 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 *multi-rank* and *multi-nullity*. 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 tensor polynomials and computation of tensor eigendecompositions, and Krylov methods. We utilize the constructs from the previous section in the context of two image processing problems, facial recognition and deblurring, in 7. We give concluding remarks in 8.

2. Preliminaries. In this section, we give the basic definitions from [10], and introduce the notation used in the rest of the paper.

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 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.

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 then

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

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

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

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

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

The following was introduced in [9, 10]:

DEFINITION 2.2. *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*

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

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.

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 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

$$(F \otimes I)\texttt{circ}(\mathcal{A})(F^* \otimes I) = \texttt{blockdiag}(\hat{A}^{(1)}, \ldots, \hat{A}^{(n)}). \tag{2.1}$$

But as the notation in the equation is intended to indicate, it is *not necessary* to form $\texttt{circ}(\mathcal{A})$ 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 $\hat{A}^{(i)} := \hat{A}(:, :, i)$. It follows from (2.1) that to compute $\mathcal{C} = \mathcal{A} * \mathcal{B}$, for example, we could[1] compute $\hat{\mathcal{A}}$ and $\hat{\mathcal{B}}$, then perform matrix multiplies of individual pairs of faces of these to give the faces of $\hat{\mathcal{C}}$, and then $\mathcal{C} = \texttt{ifft}(\mathcal{C}, [\,], 3)$.

For the remainder of the paper, the hat notation is used to indicate that we are referencing the object after having taking an FFT along the third dimension.

The following is a consequence of (2.1) that was utilized in [9] and [10] to establish the existence of certain matrix factorizations.

OBSERVATION 2.3. *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 (at least implicitly) by factoring the blocks on the block diagonal of the right-hand side of (2.1). For example, $\hat{\mathcal{A}}^{(i)} = \hat{\mathcal{Q}}^{(i)}\hat{\mathcal{R}}^{(i)}$, and then $\mathcal{Q} = \texttt{ifft}(\hat{\mathcal{Q}}, [\,], 3), \mathcal{R} = \texttt{ifft}(\hat{\mathcal{R}}, [\,], 3)$.*

The following special case of this will be important.

OBSERVATION 2.4. *Given $\mathbf{a}, \mathbf{b} \in \mathbb{R}^{1 \times 1 \times n}$, $\mathbf{a} * \mathbf{b}$ can be computed from $\texttt{ifft}(\hat{\mathbf{a}} \odot \hat{\mathbf{b}}, [\,], 3)$, where $\odot$ of two tubal-scalars means point-wise multiplication.*

For reference, we recall the conjugate symmetry of the Fourier transform. That is, if $\mathbf{v} \in \mathbb{R}^{1 \times 1 \times n}$, then $\hat{\mathbf{v}}$ satisfies the following:

[1] If $\mathcal{A}$ were sparse, it may be desirable to compute directly from the definition.

- If n is even, then for $i = 2, \ldots, n/2$, $\hat{\mathbf{v}}^{(i)} = \texttt{conj}(\hat{\mathbf{v}}^{(n-i+2)})$.
- 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 a tensor factorization: for example, to compute $\mathcal{A} = \mathcal{Q} * \mathcal{R}$, we would only need to compute individual matrix QR's for about half the faces of $\hat{\mathcal{A}}$.

Before moving on, we need a few more definitions from [10] and examples.

DEFINITION 2.5. *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.*

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

EXAMPLE 2.6. *Define $\mathcal{C} = \mathcal{A}^T * \mathcal{A}$. We could compute this via n matrix products in the Fourier domain. But, given how the tensor transpose is defined, and because of the conjugate symmetry, 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 2.7. *$\mathcal{A} \in \mathbb{R}^{m \times m \times n}$ 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. For completeness, we include here the definitions of the identity tensor and orthogonal tensor from [9].

DEFINITION 2.8. *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.*

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

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*

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

When it is not invertible, then as we will see, the kernel of the operator induced by the t-product will be nontrivial.

For convenience, we will denote by $\mathbf{e}_1$ the tubal-scalar which defines the (i, i) tubal entry of $\mathcal{I}$.

2.1. A New View on Tensors. One observation that is central to the work presented here is that tensors in $\mathbb{R}^{m \times 1 \times n}$ are simply matrices in $\mathbb{R}^{m \times n}$, oriented laterally. (See Figure 2.1.) Similarly, we can twist the matrix on the right of Figure 2.1 to a $\mathbb{R}^{m \times 1 \times n}$.

In a physical sense, if you were to stare at such a laterally oriented matrix straight on, you would only see a length-m column vector. Because these laterally oriented matrices play a crucial role in our analysis, we will therefore use the notation $\vec{\mathcal{X}}$ to denote a tensor in $\mathbb{R}^{m \times 1 \times n}$.

To go back and forth between elements in $\mathbb{R}^{m \times 1 \times n}$ and elements of $\mathbb{R}^{m \times n}$, we introduce two operators: squeeze and twist[2]. The squeeze operation works on a tensor $\vec{\mathcal{X}}$ in $\mathbb{R}^{m \times 1 \times n}$ just as it does in MATLAB:

$$X = \text{squeeze}(\vec{\mathcal{X}}) \Rightarrow X(i, j) = \vec{\mathcal{X}}(i, 1, j)$$

[2]Our thanks to Tamara Kolda for suggesting the twist function

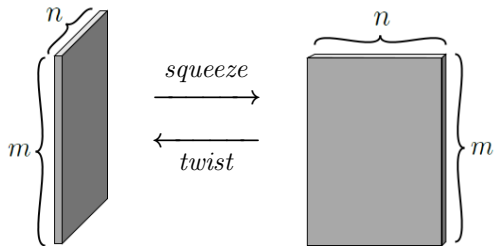

FIG. 2.1. *$m \times 1 \times n$ tensors and $m \times n$ matrices related through the* squeeze *and* twist *operations.*

whereas the twist operation is the inverse of squeeze

$$\text{twist}(\text{squeeze}(\vec{\mathcal{X}})) = \vec{\mathcal{X}}.$$

From this point on we will use $\mathbb{K}_n^m$ to denote the set of all real $m \times n$ matrices oriented as $m \times 1 \times n$ tensors. This notation is reminiscent of the notation $\mathbb{R}^m$ for vectors of length m, where the subscript in our notation has been introduced to denote the length of the tubes in the third dimension.

Because of this one-to-one correspondence between elements of $\mathbb{R}^{m \times 1 \times n}$ and elements of $\mathbb{R}^{m \times n}$, if we talk about a third-order tensor operating on a matrix, we mean formally, a third-order tensor acting on an element of $\mathbb{K}_n^m$. Likewise, the concept of orthogonal matrices is defined using the equivalent set $\mathbb{K}_n^m$, etc.

When $m = 1$, elements of $\mathbb{K}_n^m$ are just tubes of length n. For the sake of convenience, in this case we will drop the superscript when describing the collection of such objects: that is, $\mathbf{a}$ is understood to be an element of $\mathbb{K}_n$. Note that if one were viewing elements $\mathbf{a}$ directly from the front, one would see only a scalar. Furthermore, as noted in the previous section, elements of $\mathbb{K}_n$ commute. These two facts support the "tubal-scalar" naming convention used in the previous section.

Note that tubal-scalars can be thought of as the "elementary" components of tensors in the sense that $\mathcal{A} \in \mathbb{R}^{\ell \times m \times n}$ is simply an $\ell \times m$ matrix of length-n tubal-scalars. Thus, we adopt the notation that $\mathcal{A} \in \mathbb{K}_n^{\ell \times m}$, where $\mathbb{K}_n^{\ell \times m} \equiv \mathbb{R}^{\ell \times m \times n}$, but the former is more representative of $\mathcal{A}$ as a matrix of tubal-scalars. Note that $\vec{\mathcal{X}} \in \mathbb{K}_n^m$ is a vector of length-n tubal-scalars.

This view has the happy consequence of consistency with the observations in [10] that

1. multiplications, factorizations, etc. based on the t-product reduce to the standard matrix operations and factorizations when $n = 1$,
2. outer-products of matrices are well-defined
3. given $\vec{\mathcal{X}}, \vec{\mathcal{Y}} \in \mathbb{K}_n^m$, $\vec{\mathcal{X}}^T * \vec{\mathcal{Y}}$ is a scalar. We have more to say about this operation in Section 3.

2.2. Mapping Matrices to Matrices. In [10], it was noted that for $\mathcal{A} \in \mathbb{R}^{\ell \times m \times n}$, $\mathcal{X} \in \mathbb{R}^{m \times p \times n}$, $\mathcal{A} * \mathcal{X}$ defines a linear map from the space of $m \times p \times n$ tensors to the space of $\ell \times p \times n$ tensors. In the context of the matrix factorizations presented in [10], (see also [2]), the authors also noted that 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, taking $p = 1$, this means that the t-product defines a linear operator from the set of all $m \times n$ matrices to $\ell \times n$ matrices [2], where those matrices are oriented laterally. In other words, the linear operator T described by the t-product with $\mathcal{A}$ is a mapping $T : \mathbb{K}_n^m \to \mathbb{K}_n^\ell$. If $\ell = m$, T will be invertible when $\mathcal{A}$ is invertible.

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 interchangably as elements of $\mathbb{R}^{m \times n}$).

3. Inner Products, Norms and Orthogonality of Matrices . In the following, uppercase letters to describe the matrix equivalent of an $m \times 1 \times n$ tensor. In terms of our previous definitions, for example, $X := \text{squeeze}(\vec{\mathcal{X}})$.

Suppose that $\vec{\mathcal{X}}, \vec{\mathcal{Y}} \in \mathbb{K}_n^m$. Then $\mathbf{a} := \vec{\mathcal{X}}^T * \vec{\mathcal{Y}} \in \mathbb{K}_n$. In [10], the authors refer to this 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 $\langle \vec{\mathcal{X}}, \vec{\mathcal{X}} \rangle$, denoted $\langle \vec{\mathcal{X}}, \vec{\mathcal{X}} \rangle^{(1)}$, is given by $\texttt{vec}(X)^T \texttt{vec}(X)$ where X is the $m \times n$ equivalent of $\vec{\mathcal{X}}$ and the transpose and multiplication operations in the latter are over matrices. 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 otherwise, it must be non-negative.

Using the definition of $*$, $X = \text{squeeze}(\vec{\mathcal{X}})$, and using x_j to denote a column of X (with similar definitions for Y and Y_j)

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

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

We make the following definition.

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

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

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.

Now we can define a tube of angles between two matrices.

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*

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

where $|\cdot|$ is understood to be component-wise 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 effectively a length-n tube of angles (i.e. a length n vector) describing the orientation of one matrix relative to the another (see next section for more details).

In the next two examples, we show why it is important to keep track of an entire tuple of angles.

EXAMPLE 3.4. *Let* $X = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, Y = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$. *Then*

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

EXAMPLE 3.5. *Let* $X = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, Y = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$. *Then*

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

If we let $\mathcal{A}(:,1,:) = X; \mathcal{A}(:,2,:) = Y$, then in the first case, the tensor is not invertible (and thus certainly not orthogonal) while in the second, the tensor is orthogonal. This might seem odd, since in both examples, $\texttt{vec}(X) \perp \texttt{vec}(Y)$. The difference is that in "vecing", 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 angles in some sense are a measure of how close to Z^{j-1}-conjugate each of the rows of X, Y are to each other. In particular, if X, Y are such that their respective rows are all pairwise orthogonal to each other, $< \vec{\mathcal{X}}, \vec{\mathcal{Y}} >^{(1)} = 0 =< \vec{\mathcal{Y}}, \vec{\mathcal{X}} >^{(1)}$, 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 $x_1^T Z y_1 = 0 = x_2^T Z y_2$.

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

Using the bilinear form, the definition of orthogonal tensors in Definition 2.9, Observation 2.11, and the geometry, we are in a position to define what we mean by orthogonal matrices.

DEFINITION 3.6. *Given a collection of k, $m \times n$ matrices X_j, with corresponding tensors $\vec{\mathcal{X}}_j = \text{twist}(X_j)$, 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 non-zero scalar. The set is **orthonormal** *if $\alpha_i = 1$.*

This definition is consistent with our earlier definition of an orthogonality for tensors in that, $\mathcal{U} \in \mathbb{R}^{l\times m\times n}$ is an orthogonal tensor if and only if the lateral slices $\{\vec{\mathcal{U}}_1, \vec{\mathcal{U}}_2, \ldots, \vec{\mathcal{U}}_m\}$ form an orthonormal set of matrices. Note that

- Every element of an orthonormal set of matrices has unit length, $\|\vec{\mathcal{X}}_i\| = 1$.
- All components of the tubal angle between a pair of orthonormal matrices are $\pi/2$.
- In general, some of the angles will be non-zero, 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 non-zero vector the angle between v and itself is 0. Note that if $< \vec{\mathcal{X}}, \vec{\mathcal{X}} >= \alpha \mathbf{e}_1$, the first angle is 0 while the rest are $\pi/2$.

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've 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*

$$\vec{\mathcal{X}}_1 * \mathbf{c}_1 + \vec{\mathcal{X}}_2 * \mathbf{c}_2 + \cdots + \vec{\mathcal{X}}_k * \mathbf{c}_k.$$

Note that the order of multiplication is important, as in general $\mathbf{c}_j * \vec{\mathcal{X}}_j$ will not be defined unless $m = 1$. We will address this annoyance in a later section, when being able to move the tubal-scalar becomes important.

One nice feature of the t-linear combination is that it can be written in terms of a product of a third order tensor with a matrix. For example, the t-linear combination above gives

$$\sum_{i=1}^{k} \vec{\mathcal{X}}_i * \mathbf{c}_i = \mathcal{X} * \vec{\mathcal{C}},$$

where $\vec{\mathcal{C}}$ has $\mathbf{c}_1, \ldots, \mathbf{c}_k$ as its k rows and $\mathcal{X} \in \mathbb{R}^{m\times k\times n}$.

Given that we now have a concept of t-linear combination the notion of a set of building blocks for matrices, and since we know that * defines a linear map from the set of matrices to matrices, we move on to trying to characterize the range and kernel of that map.

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 non-zero entries, $\mathbf{a}$ may not be invertible[3]. According to the definition, $\mathbf{a}$ is only invertible if there exists $\mathbf{b}$ such that $\mathbf{a} * \mathbf{b} = \mathbf{b} * \mathbf{a} = \mathbf{e}_1$. But this is equivalent to saying that if $\hat{\mathbf{a}}$ is obtained by Fourier transforming $\mathbf{a}$ in the third dimension, the resulting tubal-scalar can have no zeros ($\mathbf{a}$ 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

[3]This is equivalent to the observation that $\mathbb{K}_n$ in [2] cannot form a field.

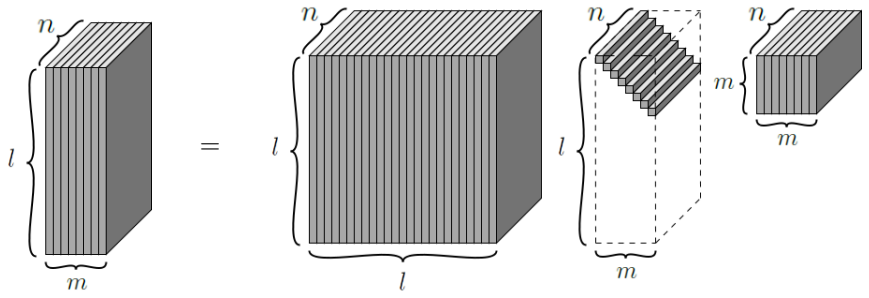

FIG. 4.1. *The t-SVD of an $l \times m \times n$ tensor.*

about the dimensionality that is not obvious directly from viewing a tensor as a matrix of tubal-scalars. That is, we want to separate tubal-scalars that are invertible from those that are not.

DEFINITION 4.2. *Suppose $\mathbf{b} \in \mathbb{K}_n$. Then its* **tubal-rank** *is the number of its non-zero Fourier coefficients.If its tubal-rank is n, we say it is* **invertible**, *if it is less than n, it is not. In particular, the tubal-rank is 0 iff $\mathbf{b} = \mathbf{0}$.*

We will use the tensor SVD, or t-SVD introduced in [10] 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 $m \times m \times n$ orthogonal $\mathcal{V}$ such that

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

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 (2.1). 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 $j \geq r_i$, $\hat{\mathbf{s}}_{jj}$ has a 0 in the ith position - i.e. $\mathbf{s}_{jj}$ has at least one zero Fourier coefficient, and therefore $\mathbf{s}_{jj}$ 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 $\mathbf{s}_{jj}$ 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 $\mathbf{s}_{jj}$ 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.

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 non-zero $\mathbf{s}_i$ are invertible. But when there are some non-zero $\mathbf{s}_i$ which are not invertible, we will have to take special care. Indeed, observe that for any $\vec{\mathcal{X}} \in \mathbb{K}_n^m$, $\mathcal{A} * \vec{\mathcal{X}} = \sum_{i=1}^{\min(\ell,m)} \vec{\mathcal{U}}_i * (\mathbf{s}_i * \vec{\mathcal{V}}_i^T * \vec{\mathcal{X}})$ where the term in parenthesis is a tubal-scalar. 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 parenthesis will have zero Fourier coefficients exactly where each $\mathbf{s}_i$ has zero Fourier coefficients.

Clearly, there are $p = min(\ell, m)$ singular tuples. Let ρ_i denote the tubal-rank of $\mathbf{s}_i$. Note that the tubal-rank cannot increase with increasing index i, so $\rho_1 \geq \rho_2 \geq \ldots \geq \rho_p$.

DEFINITION 4.3. *The multi-rank of $\mathcal{A}$ is the length-p vector $\boldsymbol{\rho} := \boldsymbol{\rho}(\mathcal{A}) = [\rho_1, \ldots, \rho_p]^T$. The multi-nullity of $\mathcal{A}$ would be the complimentary length-p vector $\boldsymbol{\eta} := \boldsymbol{\eta}(\mathcal{A}) = [n - \rho_1, \ldots, n - \rho_p]^T$.* Note that $\boldsymbol{\rho} + \boldsymbol{\eta} = [n, n, \ldots, n]^T$.

Let $\mathcal{A}$ have multi-rank $\boldsymbol{\rho} = [\overbrace{n, n, \ldots, n}^{j}, \overbrace{\rho_{j+1}, \ldots, \rho_{j+k}}^{0<\rho_j<n}, \overbrace{0, 0, \ldots, 0}^{p-k-j}]^T$, so that j singular tuples are invertible, k are non-zero but not invertible, and $p-(j+k)$ that are identically $\mathbf{0}$.

THEOREM 4.4. *The sets $R(\mathcal{A}), N(\mathcal{A})$ are given unambiguously by*

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

while

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

Proof. It is sufficient to consider two cases: when there are non-zero, non-invertible singular tuples, and when there are not.

Case I: Suppose the first $j \leq p$ singular tuples have tubal rank n, and the last $p-j$ have tubal rank 0. Recall that $\texttt{circ}(A)$ is $\ell n \times mn$. Then in this case $dim(R(\texttt{circ}(A))) = nj$ and so and in terms of tensors, this means it must be possible to find a solution to

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

for any tubal-scalars $\mathbf{c}_i$. Thus we conclude that in this case, the first j columns of $\mathcal{U}$ provide a "basis" (in the traditional sense) for $Range(\mathcal{A})$. We note that $dim(N(\texttt{circ}(A))) = nm - nj = n(m-j)$, so the last $m-j$ $\vec{\mathcal{V}}_j$ map to $\mathbf{0}$. Likewise, any t-linear combination of these maps to $\mathbf{0}$ by linearity of $*$.

Case II: Suppose that the first j, $j \neq p$, singular tuples are invertible, that singular tuples $j+1$ through $j+k$ are non-zero but not invertible, and singular tuples $j+k+1$ through p are $\mathbf{0}$ (if $k = p$, there are no zero singular tuples).

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 $\mathbf{s}_{ii}$ for those terms. In other words, we need $\mathbf{c}_i = \mathbf{d}_i * \mathbf{s}_{ii}$ for some $\mathbf{d}_i$ for $i = j+1, \ldots, j+k$.

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

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

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 $\mathbf{s}_{ii}$, for $j+1 \leq i \leq j+k$, has non-zero Fourier coefficients, and 1's where $\mathbf{s}_{ii}$ has zero Fourier coefficients. Then

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

□

This analysis shows that the number of elements in necessary to generate any element in $R(\mathcal{A})$ is the number of non-zero singular-tuples (whether or not those tuples are invertible), while the number of elements necessary to generate an arbitrary element of $N(\mathcal{A})$ is equal to $m-p$ plus the number of non-zero and non-invertible singular tuples, and so 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 $N(\mathcal{A})$. However we define dimension it should be consistent with the multi-rank of the matrix.

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

$$\dim(R(\mathcal{A})) + \dim(N(\mathcal{A})) = nm.$$

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

DEFINITION 4.5. *If $\mathcal{A} \in \mathbb{R}^{\ell \times m \times n}$, its condition number is infinite if $m > \ell$ or if the multi-nullity vector is non-zero (i.e. $\|\boldsymbol{\eta}(\mathcal{A})\|_1 > 0$). If the condition number is not infinite, then the condition vector is well defined and is given by the length n vector 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.

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 $\mathcal{P}^T = P$.*

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 $R(\mathcal{A})$.

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. That is, given a non-zero $\vec{\mathcal{X}} \in \mathbb{K}_n^m$, we need to be able to write

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

where $\|\vec{\mathcal{V}}\| = 1$ (or, in other words, $\langle \vec{\mathcal{V}}, \vec{\mathcal{V}} \rangle = \mathbf{e}_1$). This would be the same as taking $\texttt{vec}(X)$ and making it P^{j-1}-conjugate, for all $1 < j \leq n$.

Consider the following normalization algorithm. Note that $\mathbf{a}^{(j)}$ is a scalar (the jth frontal face of the $1 \times 1 \times n$ tensor $\mathbf{a}$), and 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.

$[\vec{\mathcal{V}}, \mathbf{a}]$ = Normalize($\vec{\mathcal{X}}$)
Input: $\vec{\mathcal{X}} \in \mathbb{K}_n^m \neq 0$
Output$\vec{\mathcal{V}} * \mathbf{a} = \vec{\mathcal{X}}$.
  $\vec{\mathcal{V}} = \texttt{fft}(\vec{\mathcal{X}}, [\,], 3)$
  for $j = 1 \ldots n$
    $\mathbf{a}^{(j)} = \|v_j\|_2$
    if $\mathbf{a}^{(j)} >$ tol, $\vec{\mathcal{V}}^{(j)} = \frac{1}{\mathbf{a}^{(j)}} \vec{\mathcal{V}}^{(j)}$

else $\vec{\mathcal{V}}^{(j)} = \texttt{randn}(n,1); \mathbf{a}^{(j)} = \|\vec{\mathcal{V}}^{(j)}\|_2; \vec{\mathcal{V}}^{(j)} = \frac{1}{\mathbf{a}^{(j)}}\vec{\mathcal{V}}^{(j)}; \mathbf{a}^{(j)} = 0$
endfor
$\vec{\mathcal{V}} = \texttt{ifft}(\vec{\mathcal{V}}, [\,], 3); \mathbf{a} = \texttt{ifft}(\mathbf{a}, [\,], 3)$

The following classical Gram-Schmidt algorithm 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 $\ell \times m \times n$ has orthonormal "columns" $\vec{\mathcal{Q}}_k$, $k = 1 : m$ (e.g. m orthogonal matrices), 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 CGS
Input: $\ell \times m \times n$ tensor $\mathcal{A}$, $\ell \geq m$
$[\vec{\mathcal{Q}}_1, \mathcal{R}(1,1,:)] = \text{Normalize}(\vec{\mathcal{A}}_1)$
for $i = 2 \ldots m$
$\vec{\mathcal{X}} = \vec{\mathcal{A}}_i$;
for $j = 1 \ldots i-1$
$\mathcal{R}(j,i,:) = \vec{\mathcal{Q}}_j^T * \vec{\mathcal{X}}$;
$\vec{\mathcal{X}} = \vec{\mathcal{X}} - \vec{\mathcal{Q}}_j * \mathcal{R}(j,i,:)$;
endfor
$[\vec{\mathcal{Q}}_i, \mathcal{R}(i,i,:)] = \text{Normalize}(\vec{\mathcal{X}})$;
endfor

This algorithm is consistent with the tensor QR factorization introduced in [9, 10] 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.

6. Tensor polynomials, Computation of Tensor Eigendecomposition, Krylov Subspace Methods . To motivate the work in this section, let us consider the case when the tensor $\hat{\mathcal{A}}$ 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 $\hat{\mathcal{D}}^{(i)} = \hat{D}^{(i)}$, that, upon taking the IFFT along tubes, we have an eigendecomposition [2]:

$$\mathcal{A} = \mathcal{X} * \mathcal{D} * \mathcal{X}^{-1}, \Rightarrow \mathcal{A} * \mathcal{X} = \mathcal{X} * \mathcal{D}.$$

Using Observation 2.11, we have an eigen-pair relationship,

$$\mathcal{A} * \vec{\mathcal{X}}_j = \vec{\mathcal{X}}_j * \mathbf{d}_{jj}.$$

A word of caution to the reader. Eigenvalue decomposition for tensors means different things to different researchers (see [12, 16, 18, 19], 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** $\mathbf{d}_{jj}$ and their corresponding eigenmatrices $\vec{\mathcal{X}}_j$. In our case, eigendecompositions can exist even when $m \neq n$.

The first question one might ask is when can we expect the conditions above to occur. 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 B, such an eigendecomposition exists. The next question is, what is the value in knowing this? It has been shown in

[10, 3, 6] 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.

In eigencomputation, it becomes convenient to be able to move tubal-scalar multiplication to the left.

DEFINITION 6.1. *Let* $\mathbf{a}$ *be a* $1 \times 1 \times n$ *tubal-scalar. Then*

$$\mathcal{D}_{\mathbf{a},m} := \operatorname{diag}(\mathbf{a}, \mathbf{a}, \ldots, \mathbf{a})$$

is the $m \times m \times n$ *f-diagonal tensor with a constant tubal-scalar* $\mathbf{a}$ *down the diagonal. Note that the ith frontal slice is given by* $\mathbf{a}(1,1,i)I_m$. Where the frontal dimension m is obvious, we drop the 2nd subscript.

We note that an alternative is to adopt the conventions in the paper [2], adjusted for dimension. In this work, however, we prefer not to introduce new multiplication notation and thus keep the original orientation.

Lemma: Given $\vec{\mathcal{X}} \in \mathbb{R}^{m\times 1\times n}$, and $\mathbf{a} \in \mathbb{R}^{1\times 1\times n}$,

$$\vec{\mathcal{X}} * \mathbf{a} = \mathcal{D}_{\mathbf{a}} * \vec{\mathcal{X}}.$$

This is convenient for the following reason. Consider the eigenequation $\mathcal{A} * \vec{\mathcal{X}} = \vec{\mathcal{X}} * \mathbf{d}$. We have

$$\mathcal{A} * \vec{\mathcal{X}} - \vec{\mathcal{X}} * \mathbf{d} = \mathbf{0}$$

which implies

$$\mathcal{A} * \vec{\mathcal{X}} - \mathcal{D}_{\mathbf{d}} * \vec{\mathcal{X}} = (\mathcal{A} - \mathcal{D}_{\mathbf{d}}) * \vec{\mathcal{X}} = \mathbf{0}. \tag{6.1}$$

Recall that every diagonal element of $\mathcal{D}_{\mathbf{d}}$ is the tubal-scalar $\mathbf{d}$, so the term in parentheses on the right is the result of subtracting the same tubal-scalar off the diagonal of $\mathcal{A}$.

Thus, it would seem that to get a handle on eigentuples, we need to find $\mathbf{d}$ such that $\mathcal{A} - \mathcal{D}_{\mathbf{d}}$ is not invertible. This brings us to the notion of determinants for tensors. Our convention of thinking of $\mathcal{A}$ as a matrix as tubal-scalars for entries leads us to the following definition.

DEFINITION 6.2. *Let* $\mathcal{A} \in \mathbb{K}^{2\times 2\times n}$. *Then*

$$\mathtt{det}(\mathcal{A}) := |\mathcal{A}| = \mathbf{a}_{11} * \mathbf{a}_{22} - \mathbf{a}_{12} * \mathbf{a}_{21} \in \mathbb{K}_n.$$

Note that the output of the determinant of a $2 \times 2 \times n$ tensor is a tubal-scalar, and not a scalar (unless $n = 1$).

Now it is easy to see that expansion by minors in the usual way (with scalar multiplication replaced by $*$) is well defined for $\mathcal{A} \in \mathbb{K}_n^{m\times m}$.

EXAMPLE 6.3. *Let* $\mathcal{A} \in \mathbb{R}^{3\times 3\times n}$. *Expanding by minors on the first row, we have*

$$\mathtt{det}(\mathcal{A}) = \mathbf{a}_{11} * \mathtt{det}\left(\begin{bmatrix} \mathbf{a}_{22} & \mathbf{a}_{23} \\ \mathbf{a}_{32} & \mathbf{a}_{33} \end{bmatrix}\right) - \mathbf{a}_{12} * \mathtt{det}\left(\begin{bmatrix} \mathbf{a}_{21} & \mathbf{a}_{23} \\ \mathbf{a}_{31} & \mathbf{a}_{33} \end{bmatrix}\right) + \mathbf{a}_{13} * \mathtt{det}\left(\begin{bmatrix} \mathbf{a}_{21} & \mathbf{a}_{22} \\ \mathbf{a}_{31} & \mathbf{a}_{32} \end{bmatrix}\right).$$

In order to connect determinants with invertibility of tensors, it is again necessary to appeal to the Fourier domain.

LEMMA 6.4. *Let $\mathcal{A} \in \mathbb{K}_n^{m\times m}$. Then* $\texttt{det}(\mathcal{A})$ *can be obtained by computing matrix determinants of each frontal face of $\hat{\mathcal{A}}$, putting each results into the corresponding face of a tubal-scalar $\hat{\mathbf{z}}$ and taking the inverse Fourier transform of the result.*

Proof. We prove this for $m = 2$, extension to the general case is obvious. We have

$$\texttt{det}(\mathcal{A}) = \mathbf{a}_{11} * \mathbf{a}_{22} - \mathbf{a}_{12} * \mathbf{a}_{21} \tag{6.2}$$

$$= \texttt{ifft}(\hat{\mathbf{a}}_{11} \odot \hat{\mathbf{a}}_{22}, [\,], 3) - \texttt{ifft}(\hat{\mathbf{a}}_{12} \odot \hat{\mathbf{a}}_{21}, [\,], 3) \tag{6.3}$$

$$= \texttt{ifft}\left((\hat{\mathbf{a}}_{11} \odot \hat{\mathbf{a}}_{22}) - (\hat{\mathbf{a}}_{12} \odot \hat{\mathbf{a}}_{21})[\,], 3\right) \tag{6.4}$$

where the last equality follows by linearity of the discrete inverse Fourier Transform. But the tubal-scalar in the interior has as its jth element the determinant of the jth face, namely $\hat{\mathbf{a}}_{11}^{(j)}\hat{\mathbf{a}}_{22}^{(j)} - \hat{\mathbf{a}}_{12}^{(j)}\hat{\mathbf{a}}_{21}^{(j)}$, and the result is proved. ∎

We need one more important lemma.

LEMMA 6.5. *Given $\mathcal{A}, \mathcal{B} \in \mathbb{K}_n^{m\times m}$,* $\texttt{det}(\mathcal{A} * \mathcal{B}) = \texttt{det}(\mathcal{A}) * \texttt{det}(\mathcal{B})$.

Proof. Define $\mathcal{C} = \mathcal{A} * \mathcal{B}$. We know the entries in $\mathcal{C}$ can be computed by first forming $\hat{\mathcal{A}}, \hat{\mathcal{B}}$, then computing pairs of matrix products in the Fourier domain: $\hat{\mathcal{C}}^{(j)} = \hat{\mathcal{A}}^{(j)}\hat{\mathcal{B}}^{(j)}$. But $\texttt{det}(\hat{\mathcal{C}}^{(j)}) = \texttt{det}(\hat{\mathcal{A}}^{(j)})\texttt{det}(\hat{\mathcal{B}}^{(j)})$, so using the preceding lemma and this equality, the jth Fourier coefficient of $\texttt{det}(\mathcal{C})$ is the product of the jth Fourier coefficients of $\texttt{det}(\mathcal{A})$ and $\texttt{det}(\mathcal{B})$. But by Fact 1, this is exactly the right-hand side of the equality we were trying to prove. ∎

Finally, we are in a position to show that if $\mathcal{Q}$ is an orthogonal tensor, $\texttt{det}(\mathcal{Q})$ can have no zero Fourier coefficients.

LEMMA 6.6. *Let $\mathcal{Q} \in \mathbb{K}_n^{m\times m}$ be an orthogonal tensor. Then* $\texttt{det}(\mathcal{Q})$ *has no zero Fourier coefficients.*

Proof. By the definition, $\mathcal{Q}^T * \mathcal{Q} = \mathcal{I}$, and by the preceding lemma, $\texttt{det}(\mathcal{Q}^T * \mathcal{Q}) = \texttt{det}(\mathcal{Q})^T * \texttt{det}(\mathcal{Q}) = \texttt{det}(\mathcal{I})$. But it is easily shown that $\texttt{det}(\mathcal{I}) = \mathbf{e}_1$, which has no zero Fourier coefficients. Because of our Fact, it follows that neither $\texttt{det}(\mathcal{Q})$ nor $\texttt{det}(\mathcal{Q}^T)$ can have zero Fourier coefficients, or we would have a contradiction. ∎

Is the determinant linked to the invertibility of the tensor? To see that this is so, consider first an f-uppertriangular $\mathcal{A} \in \mathbb{K}_n^{m\times m}$. This will not be invertible if any diagonal element $\mathbf{a}_{jj}$ has at least one zero Fourier coefficient. The ith Fourier coefficient of $\texttt{det}(\mathcal{A})$ is 0 iff the ith coefficient of any $\hat{\mathbf{a}}_{jj}$ is 0. If that happens, $\rho_j < n$, and we know that $N(\mathcal{A})$ is not empty and thus $\mathcal{A}$ is not invertible. We can move to the general case using the t-QR of $\mathcal{A}$ and the fact that if $\mathcal{A} = \mathcal{Q} * \mathcal{R}$, $\texttt{det}(\mathcal{A}) = \texttt{det}(\mathcal{Q}) * \texttt{det}(\mathcal{R})$, and $\texttt{det}(\mathcal{Q})$ has no zero Fourier coefficients.

Thus, if we want to find an eigentuple for $\mathcal{A}$, given (6.1) we should of course look for where $\texttt{det}(\mathcal{A} - \mathcal{D}_{\mathbf{d}}) = \mathbf{0}$.

It follows that

$$\texttt{det}(\mathcal{A} - \mathcal{D}_{\mathbf{d}})$$

must be a polynomial (note scalars have been replaced by tubal-scalars) of degree m in $\mathbf{d}$, call it $p(\mathbf{d})$. What constitutes a zero of this polynomial? Going back to the test case of an f-uppertriangular $\mathcal{A} \in \mathbb{K}^{2\times 2\times n}$, the characteristic equation is obviously of the form

$$(\mathbf{a}_{11} - \mathbf{d}) * (\mathbf{a}_{22} - \mathbf{d}) = \mathbf{0}. \tag{6.5}$$

Clearly, there are roots $\mathbf{a}_{11}, \mathbf{a}_{22}$. But, as we illustrate below, there are actually may be many more roots in $\mathbb{K}_n$ of this equation!

THEOREM 6.7. *When n is even, there may be as many as* $\binom{n/2+1}{2}$ *solutions in* $\mathbb{K}_n$ *to (6.5). When n is odd, there may be as many as* $\binom{(n+1)/2}{2}$.

Proof. Taking Fourier transforms along the tubes and the equating componentwise gives the n scalar equations

$$(\hat{\mathbf{a}}_{11} - \hat{\mathbf{d}})^{(i)}(\hat{\mathbf{a}}_{22} - \hat{\mathbf{d}})^{(i)} = 0, \qquad i = 1, \ldots, n.$$

Each scalar equation can be satisfied either by $\hat{\mathbf{d}}^{(i)} = \hat{\mathcal{A}}_{11}^{(i)}$ or $\hat{\mathbf{d}}^{(i)} = \hat{\mathcal{A}}_{22}^{(i)}$. In order for the inverse FFT of $\hat{\mathbf{d}}$ to be real, we must maintain conjugate symmetry. Thus, when the first $n/2+1$ terms (in the even case) or $(n+1)/2$ (in the odd case) terms are selected, the remaining components in $\hat{\mathbf{d}}$ are determined. □

The preceding theorem indicates that *generalizing matrix eigenvalue algorithms, such as the power iteration, is not completely straightforward* and will require a little more investigation.

6.1. Powers of Tensors. If we want to investigate the extension of the power iteration, QR 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 $\mathcal{A}^k := \underbrace{\mathcal{A} * \mathcal{A} \cdots * \mathcal{A}}_{\text{k times}}$. The result can be computed using

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

Thus, to investigate what happens with a tensor power, it suffices to study these individual matrix powers.

Suppose that $\hat{A}^{(i)}$ is diagonalizable, with $\hat{A}^{(i)} = Z^{(i)}\Lambda^{(i)}(Z^{(i)})^{-1}$. For convenience, suppose entries in each of the diagonal matrices $\Lambda^{(i)}$ are ordered in decreasing magnitude so that the 1,1 entry of $\Lambda^{(i)}$, denoted as $\lambda_1^{(i)}$, has the property $|\lambda^{(i)}|_1 > |\lambda^{(i)}|_2 \geq \cdots \geq |\lambda^{(i)}|_m$. Clearly, $(\hat{A}^{(i)})^k = Z^{(i)}(\Lambda^{(i)})^k(Z^{(i)})^{-1}$.

Under these assumptions, it is clear that $\mathcal{A}$ is f-diagonalizable: $\mathcal{A} = \mathcal{V} * \mathcal{D} * \mathcal{V}^{-1}$, where $\mathcal{V}$ comes from inverse FFTs along tubes of the tensor with faces $Z^{(i)}$, $\mathcal{D}$ comes from inverse FFTs along tubes of the tensor with faces $\Lambda^{(i)}$.

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

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

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 eigen-tuple corresponding to the largest magnitude entries of each $\Lambda^{(i)}$. But, without the proper normalization of the candididate eigenmatrix at each iteration, this will not happen! The easiest way to see this is to recall (2.1) 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, 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 $\hat{A}^{(i)}$.

Algorithm Power Iteration
Input: $m \times m \times n$ tensor $\mathcal{A}$
Output: $\mathbf{d}, \vec{\mathcal{V}}$ such that $\mathcal{A} * \vec{\mathcal{V}} \approx \vec{\mathcal{V}} * \mathbf{d}$.
$\vec{\mathcal{V}} = \texttt{randn}(m, 1, n)$;
$\vec{\mathcal{V}} = \text{normalize}(\vec{\mathcal{V}})$;
for i=1...
$\vec{\mathcal{V}} = \mathcal{A} * \vec{\mathcal{V}}$;
$\vec{\mathcal{V}} = \text{normalize}(\vec{\mathcal{V}})$;
$\mathbf{d} = \vec{\mathcal{V}}^T * (\mathcal{A} * \vec{\mathcal{V}})$;
endfor

6.2. Similarity Transforms and QR Iteration. Let $\mathcal{Q}$ be an invertible tensor. Then if $\mathcal{A} = \mathcal{Q} * \mathcal{B} * \mathcal{Q}^{-1}$, we say $\mathcal{A}, \mathcal{B}$ are similar. Note that this means that $\mathcal{A}, \mathcal{B}$ have the same eigentuples, since

$$\mathcal{A} * \vec{\mathcal{V}}_i = \vec{\mathcal{V}}_i * \mathbf{d}_{ii} \Rightarrow (\mathcal{Q} * \mathcal{A} * \mathcal{Q}^{-1}) * (\mathcal{Q} * \vec{\mathcal{V}}_i) = (\mathcal{Q} * \vec{\mathcal{V}}_i) * \mathbf{d}_{ii}.$$

So, the standard QR iteration for tensors works exactly as it does for matrices in the sense that it produces a sequence of tensors that are unitarily similar to $\mathcal{A}$. It is easy to see that $\mathcal{A}$ could first be reduced to f-upper Hessenberg or f-tridagional (in the facewise-symmetric case), and then QR Iteration is applied after the direct factorization step. The following algorithm makes use of the t-QR factorization from [9] (also described in Observation 2.3).

Algorithm QR Iteration
Input: $m \times m \times n$ tensor $\mathcal{A}$
For i=1...
$[\mathcal{Q}, \mathcal{R}] = \text{t-QR}(\mathcal{A})$;
$\mathcal{A} = \mathcal{R} * \mathcal{Q}$;
endfor

It is straightforward to show this iteration is equivalent to doing QR iteration on each of the subblocks of the block diagonal matrix in Fourier space and so convergence (to f-upper triangular, f-diagonal in the symmetric case) is assured when the appropriate conditions for convergence are met for each of the blocks in Fourier space. Convergence (theoretical) should be to the factorization $\mathcal{A} = \mathcal{V} * \mathcal{D} * \mathcal{V}^{-1}$ where $\mathcal{D}$ is the one you get when the eigenvalues are arranged in descending magnitude order.

In future work we will investigate the implications of adding tubal-scalar shifts as well as behavior of the algorithm in finite precision arithmetic.

6.3. Krylov Subspace Methods. The Krylov tensor generated by $\mathcal{A}$ and $\vec{\mathcal{B}}$ comprised of k lateral slices is defined according to

$$\mathcal{K}_k := [\vec{\mathcal{B}}, \mathcal{A} * \vec{\mathcal{B}}, \mathcal{A}^2 * \vec{\mathcal{B}}, \cdots, \mathcal{A}^{k-1} * \vec{\mathcal{B}}].$$

We assume that $k \leq \min(\ell, m)$ and that $\vec{\mathcal{B}}$ has an invertible normalization tubal scalar, and 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).

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 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. With this in mind, we give versions of the Symmetric Lanczos iteration (compare to the matrix version, [25, Alg 36.1]) which would be applicable when $\mathcal{A}$ is symmetric positive definite, and bidiagonalization (compare to the matrix version as in [25]).

Algorithm Symmetric Lanczos Iteration
Input: $m \times m \times n$ tensor $\mathcal{A}$
Input: $m \times 1 \times n$ non-zero tensor $\vec{\mathcal{B}}$
$\vec{\mathcal{Q}}_0 = 0$.
$[\vec{\mathcal{Q}}_1, \mathbf{z}_0] = \text{normalize}(\vec{\mathcal{B}})$
For i=1...
$\vec{\mathcal{V}} = \mathcal{A} * \vec{\mathcal{Q}}_i$
$\mathbf{c}_n = \vec{\mathcal{Q}}_n^T * \vec{\mathcal{V}}$
$\vec{\mathcal{V}} = \vec{\mathcal{V}} - \vec{\mathcal{Q}}_{i-1} * \mathbf{z}_{i-1} - \vec{\mathcal{Q}}_i * \mathbf{c}_i$
$[\vec{\mathcal{Q}}_{i+1}, \mathbf{z}_i] = \text{normalize}(\vec{\mathcal{V}})$
endfor

Since we have assumed for simplicity that the algorithm does not breakdown, in exact arithmetic taken to k steps this would produce $\mathcal{Q}_k^T * \mathcal{A} * \mathcal{Q}_k = \mathcal{T}$ where

$$\mathcal{T} = \begin{bmatrix} \mathbf{c}_1 & \mathbf{z}_1 & 0 & 0 & 0 \\ \mathbf{z}_1 & \mathbf{c}_2 & \mathbf{z}_2 & \cdots & 0 \\ 0 & \mathbf{z}_2 & \mathbf{c}_2 & \ddots & \\ \vdots & \ddots & \ddots & \ddots & \mathbf{z}_k \\ 0 & \cdots & 0 & \mathbf{z}_k & \mathbf{c}_k \end{bmatrix},$$

is an f-tridiagonal $k+1 \times k \times n$ 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 $O(k)$ FFT's and IFFT's 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) can likewise be extended to the tensor case:

Algorithm Golub-Kahan Iterative Bidiagonalization

Input: $\ell \times m \times n$ tensor $\mathcal{A}$
Input: $m \times 1 \times n$ non-zero tensor $\vec{\mathcal{B}}$
$\vec{\mathcal{Q}}_0 = 0$.
$[\vec{\mathcal{Q}}_1, \mathbf{z}_1] = \text{normalize}(\vec{\mathcal{B}})$ with $\mathbf{z}_1$ invertible
For i=1...
$\vec{\mathcal{W}}_i = \mathcal{A}^T * \vec{\mathcal{Q}}_i - \vec{\mathcal{W}}_{i-1} * \mathbf{z}_i$
$[\vec{\mathcal{W}}_i, \mathbf{c}_i] = \text{normalize}(\vec{\mathcal{W}}_i)$
$\vec{\mathcal{Q}}_{i+1} = \mathcal{A} * \vec{\mathcal{W}}_i - \vec{\mathcal{Q}}_i * \mathbf{c}_i$
$[\vec{\mathcal{Q}}_{i+1}, \mathbf{z}_{i+1}] = \text{normalize}(\vec{\mathcal{Q}}_{i+1})$
endfor

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

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

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 FFT's, and $O(n/2)$ SVDs of bidiagonal matrices[4].

The methods as introduced clearly refer to producing approximate tensor eigen decomposition and a tensor SVD. 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

$$\mathcal{A} * \vec{\mathcal{X}} = \vec{\mathcal{B}}$$

or

$$\min_{\vec{\mathcal{X}}} \|\mathcal{A} * \vec{\mathcal{X}} - \vec{\mathcal{B}}\|,$$

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 perhaps $\mathcal{A}, \vec{\mathcal{B}}$ 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 alternate view would be to develop a CG algorithm directly from the Symmetric Lanczos iteration. We present the CG algorithm here (compare to [25, Alg. 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

[4]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

3), so it is not quite appropriate to think of tensor CG as quite akin to minimizing the A-norm of the error. On the other hand, 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 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 t-CG
Input: $m \times m \times n$ positive definite tensor $\mathcal{A}$
Input: $m \times 1 \times n$ non-zero tensor $\vec{\mathcal{B}}$
OutputEstimate $\vec{\mathcal{X}}$, $\mathcal{A}$-conjugate $\vec{\mathcal{P}}_i$, orthogonal $\vec{\mathcal{R}}_i$.
$\vec{\mathcal{X}}_0 = \vec{r}$.
$[\vec{\mathcal{R}}_0, \mathbf{a}] = \vec{\mathcal{B}}, \vec{\mathcal{P}}_0 = \vec{\mathcal{R}}_0$.
For $i = 1, \ldots,$
$\mathbf{c} = (\vec{\mathcal{P}}^T * \mathcal{A} * \vec{\mathcal{P}})^{-1} * (\vec{\mathcal{R}}^T * \vec{\mathcal{R}})$
$\vec{\mathcal{X}}_i = \vec{\mathcal{X}}_{i-1} + \vec{\mathcal{P}}_{i-1} * \mathbf{c}$
$\vec{\mathcal{R}}_i = \vec{\mathcal{R}}_{i-1} - \mathcal{A} * (\vec{\mathcal{P}}_i * \mathbf{c})$
$\mathbf{d} = (\vec{\mathcal{R}}_{i-1}^T * \vec{\mathcal{R}}_{i-1})^{-1} * (\vec{\mathcal{R}}_i^T * \vec{\mathcal{R}}_i)$
$\vec{\mathcal{P}}_i = \vec{\mathcal{R}}_i + \vec{\mathcal{P}}_{i-1} * \mathbf{d}$
endfor
$\vec{\mathcal{X}} = \vec{\mathcal{X}} * \mathbf{a}$.

7. Numerical Examples. In this section we give two examples illustrating the potential of some 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, $b := \texttt{vec}B$ 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 A has significant structure. For purposes of illustrating on Krylov solver algorithm, we will assume that K is block circulant with Toeplitz blocks.

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

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

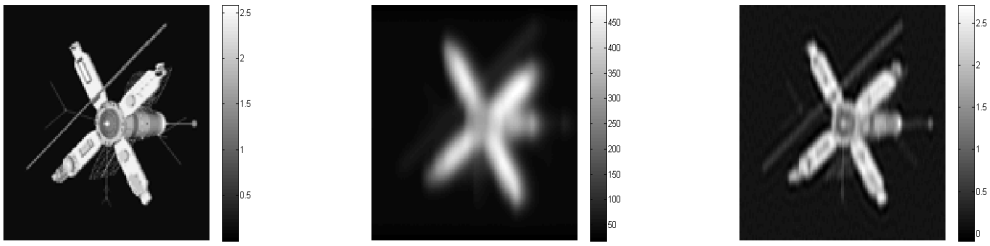

FIG. 7.1. *True image (left), blurred noisy image (middle), reconstruction after 42 iterations our tensor CG algorithm on the regularized normal equations with* $\lambda^2 = 2$.

The assumed structure on K ensures that $K^TK + \lambda^2 I$ is still a block circulant matrix. Given this structure, as discussed in [10], 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 $\vec{\mathcal{X}} = \text{twist}(X)$, $\vec{\mathcal{B}} = \text{twist}(B)$, and $\mathbf{d} = \lambda^2 \mathbf{e}_1$.

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 illustration. 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 [10] (specifically, K here is equivalent to $\tilde{A}$ in that paper), and folded the first block column to obtain $\mathcal{A}$. The true image X is given in Figure 7.1. The blurred, noisy image is created first by computing $\mathcal{A} * \vec{\mathcal{X}}$, where $\vec{\mathcal{X}} = \text{twist}(X)$, then adding random Gaussian noise with a noise level of 0.1 percent. The blurred noisy image is shown in Figure 7.1. We set $\lambda^2 = 2$ and found that after 42 iterations, the minimum relative error between $\vec{\mathcal{X}}$ and the image estimate was 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 Facial Recognition. In this experiment, we are provided with a set of 200 gray-scale training images of size 112×92. The images are from the AT&T training set [1] and consist of photos of 40 individuals under various conditions. We construct a normalized database of these images as follows. From each image we subtract the mean of all the images. Each of the adjusted images are placed as lateral slices into a tensor $\mathcal{A}$.

In a real facial recognition scenario, if a new (normalized) image $\vec{\mathcal{Y}}$ becomes available, the size of the database is too large to manipulate to determine the identity

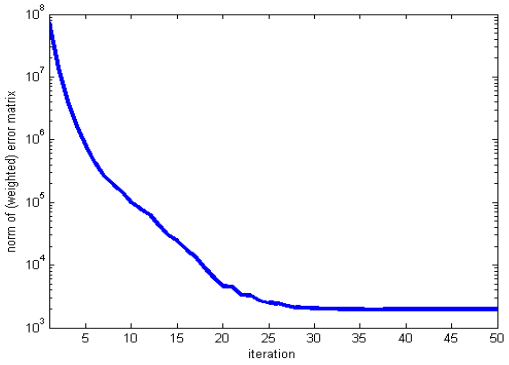

FIG. 7.2. *Estimate of the norm of the error as a function of iteration.*

of the person in the photo. Thus, the desire is to compress the database, and one way to do this would be to truncate the t-SVD of $\mathcal{A}$ [3, 6]. The implication is that that first k lateral slices of $\mathcal{U}$ contain enough information about the photos in the database that computing the projection coefficients of the projection onto the t-span of the first k columns of $\mathcal{U}$ should reveal the identity of the individual. In other words, since $\vec{\mathcal{A}}_j \approx \mathcal{U}(:,1:k,:) \underbrace{(\mathcal{U}(:,1:k,:)^T * \vec{\mathcal{A}}_j)}_{\vec{\mathcal{C}}_j}$, we can compare $\vec{\mathcal{C}}_j$ with $\mathcal{U}(:,1:k,:)^T * \vec{\mathcal{Y}}$ to answer the identity question.

Furthermore, if the person is in the database and k is sufficiently large, we expect that $\vec{\mathcal{Y}} \approx \underbrace{\mathcal{U}(:,1:k,:) * \mathcal{U}(:,1:k,:)^T}_{\mathcal{P}} * \vec{\mathcal{Y}}$, where $\mathcal{P}$ is an orthogonal projector. Taking the image computed on the right of this equation and adding the mean back in we get what we refer to as the reconstructed image.

In this example, we compute an estimate of $\mathcal{U}(:,1:k,:)$ for $k = 20$, using the tensor GK iterative bidiagonalization algorithm from the previous section. In Figure 7.3, we show the original image (leftmost), the reconstruction using the first k terms of the $\mathcal{U}$ from the t-SVD, the reconstruction using the estimate based on the GK iterative bidiagonalization routine from the previous section, and the reconstruction from the well-known eigenfaces (aka PCA) approach [27] with the same number of basis vectors. In the eigenfaces approach, the database is a matrix arrived at by unfolding each of our columns of $\mathcal{A}$, the basis vectors are traditional singular vectors, projection is done by applying to the vectorized form of $\vec{\mathcal{Y}}$, then reshaping. We believe our tensor-based approach[5] is superior because it enables use to treat the images as the two dimensional objects that they are. In [9], it was shown that if $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$, then summing each tensor in the third dimension we obtain a matrix SVD of the summed $\mathcal{A}$. In this plot, we show the result of summing $\mathcal{S}$ in the third dimension compared with summing the estimated leading submatrix of $\mathcal{S}$ along the third dimension. There is good agreement for small k. For large 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

[5] There is also a well-known tensor-based approach to this problem called Tensor-faces [29, 28, 30]. That work is based on a higher-order representation, but the decomposition used to obtain projections is significantly different than ours. A comparsion of our methods to that approach is the subject of current research.

FIG. 7.3. *Reconstructions of certain views of individuals in the AT&T database. Left is using 20 columns of the t-SVD $\mathcal{U}$ matrix; middle is using 20 columns of the estimates of $\mathcal{U}$ via our bidiagonalization; Right is using 20 terms of traditional PCA. Note the similarity between the leftmost pair of reconstructions, and their superiority over the more traditional PCA approach.*

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 [2] 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 2D 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, QR, and 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

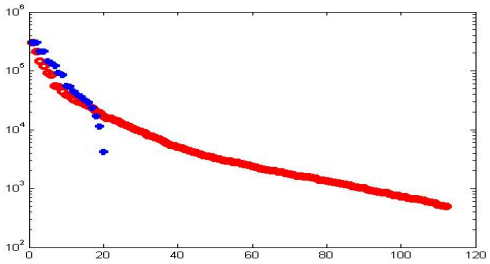

FIG. 7.4. *Comparison of* $\texttt{sum}(\mathcal{S}, [\,], 3)$ *with the estimate computed via our bidiagonalization routine.*

where the framework presented here will indeed prove valuable.

REFERENCES

[1] C. AT&T LABORATORIES, *The database of faces*. http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

[2] K. BRAMAN, *Third-order tensors as linear operators on a space of matrices*, Linear Algebra and its Applications, (2010), pp. 1241–1253.

[3] K. BRAMAN AND R. HOOVER, *Tensor decomposition and application in signal and image processing*. for abstract, http://www.siam.org/meetings/an10/An10abstracts.pdf, page 58. Invited Minisymposium Presentation, 2010 SIAM Annual Meeting (MS35).

[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] 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.

[6] N. HAO AND M. KILMER, *Tensor-svd with applications in image processing*. for abstract, http://www.siam.org/meetings/an10/An10abstracts.pdf, page 58. Invited Minisymposium Presentation, 2010SIAM Annual Meeting (MS35).

[7] 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.

[8] 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 Transactions on Image Processing, 14 (2005), pp. 884–889.

[9] 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 Technical Report TR-2008-4, Tufts University, Department of Computer Science, 2008.

[10] M. E. KILMER AND C. D. MARTIN, *Factorization strategies for third-order tensors*, Linear Algebra and its Applications, Special Issue in Honor of G. W. Stewart's 70th birthday (2010). In press. DOI: 10.1016/j.laa.2010.09.020.

[11] T. KOLDA AND B. BADER, *Tensor decompositions and applications*, SIAM Review, 51 (2009), pp. 455–500.

[12] T. G. KOLDA AND J. R. MAYO, *Shifted power method for computing tensor eigenpairs*. arXiv:1007.1267v1 [math.NA], July 2010.

[13] P. KROONENBERG, *Three-mode principal component analysis: Theory and applications*, DSWO Press, Leiden, 1983.

[14] L. D. LATHAUWER, B. D. MOOR, , AND J. VANDEWALLE, *A multilinear singular value decomposition*, SIAM Journal of Matrix Analysis and Applications, 21 (2000), pp. 1253–1278.

[15] L. D. LATHAUWER AND B. D. MOOR, *From matrix to tensor: Multilinear algebra and signal processing*, in Mathematics in Signal Processing IV, J. McWhirter and e. I. Proudler, eds., Clarendon Press, Oxford, UK, 1998, pp. 1–15.

[16] L.-H. LIM, *Singular values and eigenvalues of tensors: A variational approach*, in in CAMSA P'05: Proceeding of the IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, 2005, pp. 129–132.

[17] J. NAGY AND M. KILMER, *Kronecker product approximation for preconditioning in three-dimensional imaging applications*, IEEE Trans. Image Proc., 15 (2006), pp. 604–613.

[18] 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.
[19] L. Qi, *Eigenvalues and invariants of tensors*, Journal of Mathemathematical Analysis and Applications, 325 (2007), pp. 1363–1367.
[20] M. Rezghi and L. Eldèn, *Diagonalization of tensors with circulant structure*, Linear Algebra and its Applications, Special Issue in Honor of G. W. Stewart's 70th birthday (2010). In Press. Available On-line Oct. 2010.
[21] B. Savas and L. Eldén, *Krylov subspace methods for tensor computations*, Tech. Report LITH-MAT-R-2009-02-SE, Department of Mathematics, Linkpings Universitet, 2009.
[22] B. Savas and L. Eldén, *Krylov-type methods for tensor computations*, Submitted to Linear Algebra and its Applications, (2010).
[23] N. Sidiropoulos, R. Bro, and G. Giannakis, *Parallel factor analysis in sensor array processing*, IEEE Transactions on Signal Processing, 48 (2000), pp. 2377–2388.
[24] A. Smilde, R. Bro, and P. Geladi, *Multi-way Analysis: Applications in the Chemical Sciences*, Wiley, 2004.
[25] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM Press, 1997.
[26] L. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.
[27] M. Turk and A. Pentland, *Face recognition using eigenfaces*, in Proc. IEEE Conference on Computer Vision and Pattern Recognition, 1991, p. 586591.
[28] M. Vasilescu and D. Terzopoulos, *Multilinear analysis of image ensembles: Tensorfaces*, Proceedings of the 7th European Conference on Computer Vision ECCV 2002, (2002), pp. 447–460. Vol. 2350 of Lecture Notes in Computer Science.
[29] ———, *Multilinear image analysis for face recognition*, Proceedings of the International Conference on Pattern Recognition ICPR 2002, 2 (2002), pp. 511–514. Quebec City, Canada.
[30] ———, *Multilinear subspace analysis of image ensembles*, Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR 2003, (2003), pp. 93–99.