Updating the multilinear UTV decomposition

Michiel Vandecappelle*†, *Student member, IEEE,*, and Lieven De Lathauwer*†, *Fellow, IEEE*

***Abstract*—Low multilinear rank approximation (LMLRA) is a basic tool for compressing a tensor into a more compact form, while preserving most of its information. An LMLRA of an Nth order tensor consists of N factor matrices which are combined using a core tensor of smaller dimensions than the original tensor. When the tensor is non-static, for example when over time new tensor slices are being added along a certain mode, one can expect that the LMLRA of the tensor changes, but that an approximation of the new tensor can be derived from the previous one. By updating the factor matrices and the core tensor in an efficient way when a new tensor slice is added, this can indeed be achieved. In this paper, a new method is proposed to track the factor matrices in both the evolving and non-evolving modes, leading to efficient and accurate updates of the LMLRA. We track a truncated multilinear UTV decomposition (TMLUTVD), which is a tensor decomposition that is multilinear rank-revealing, yet less expensive to compute than the popular multilinear singular value decomposition. We derive accuracy bounds for the tracked TMLUTVD and give strategies to speed up its computation. Experiments show that the TMLUTVD tracking method works well in practice, even if the mode-n principal subspaces change drastically between updates. As an illustration of the possibilities of the technique, we extend the use of LMLRA in a compression step, prior to the computation of a canonical polyadic decomposition (CPD), to the setting of updating and tracking.**

***Index Terms*—Tensors, LMLRA, subspace tracking**

I. Introduction

Tensors have become an important tool in many applications in machine learning [1] and signal processing [2]. These higher-order generalizations of vectors and matrices are particularly powerful for the storage and analysis of higher-order data, as they preserve their inherent higher-order structure. The number of entries in a tensor grows exponentially with the tensor order, so tensor methods can easily strain the available resources. To extract the meaningful information from a tensor, different tensor decompositions have been proposed, enabling feature extraction, dimensionality reduction or compression of tensor data. Some of the most commonly used tensor decompositions are the canonical polyadic decomposition (CPD) [3] and the low multilinear rank approximation (LMLRA) [4]. A particular instance of an LMLRA is given by the truncated multilinear singular value Decomposition (TMLSVD) [5], which can be computed algebraically through N SVDs and a multilinear projection. Although truncation of the MLSVD does not offer the optimal LMLRA, its approximation error is often very close to minimal [5], [6]. As a higher-order extension of the matrix SVD, the MLSVD inherits the good numerical properties of the SVD, making it a powerful tool for signal subspace estimation, orthogonal compression, multilinear principal component analysis, etc. Some notable examples can be found in sensor arrays, denoising, multi-user detection, time delay estimation, image segmentation, speech enhancement, magnetic resonance spectroscopy, smart antennas and radar systems [1], [2], [7], [8]. A TMLSVD of a tensor is obtained by computing SVDs of its tensor unfoldings and trimming all but the multilinear singular vectors corresponding to the largest multilinear singular values. If required, the core tensor of the decomposition is then constructed afterwards. As the multilinear singular values are sorted by their magnitude, truncation to any multilinear rank provides a an approximation of the tensor that is close to optimal. In real-time applications, new tensor data can become available over time and old tensor values can become outdated; see Figure 1 for the situation where new matrix slices are appended along the third mode. Because of time or memory constraints, recomputing a full LMLRA when only a small subset of the tensor entries are added or modified, is often impossible. Fortunately, methods exist that can update the old tensor decomposition with the new information quickly and accurately. When tracking a noisy low-rank tensor with a TMLSVD, one is often mainly interested in extracting the principal *subspace* of the tensor in every mode and less in extracting the individual singular values: all multilinear singular vectors in the principal subspaces are retained, while those corresponding to the noise are trimmed. In such cases, one can choose to track for every mode a rank-revealing decomposition that is cheaper than the SVD [9]. For example, we may consider the rank-revealing UTV (RRUTV) decomposition, wherein the $\mathbf{U}$ and $\mathbf{V}$ matrices are orthogonal and the $\mathbf{T}$ matrix is triangular (either lower triangular for the ULV decomposition or upper triangular for the URV decomposition) instead of diagonal as in the SVD. Below, we use the term UTV decomposition when there is no need to distinguish between ULV and URV. A UTV decomposition provides a basis for the principal subspace that is less expensive to compute than the basis obtained by the SVD [10], [11]. The bases for the principal subspaces in every mode can then be used as factor matrices for a truncated multilinear UTV decomposition [12] (TMLUTVD) instead of a TMLSVD. If necessary, the core tensor can be computed afterwards through a multilinear projection. An

Funding: Michiel Vandecappelle was partially supported by an SB Grant (1S66617N) from the Research Foundation—Flanders (FWO). Research furthermore supported by: (1) Flemish Government: by Fonds de la Recherche Scientifique–FNRS and the Fonds Wetenschappelijk Onderzoek–Vlaanderen under EOS project no 30468160 (SeLMA) and by Artificiële intelligentie (AI) Vlaanderen (3E190661); (2) KU Leuven Internal Funds C16/15/059 and ID-N project no 3E190402; (3) Leuven Institute for Artificial Intelligence (Leuven.ai).

*KU Leuven, Dept. of Electrical Engineering ESAT/STADIUS, Kasteelpark Arenberg 10, bus 2446, B-3001 Leuven, Belgium

† KU Leuven – Kulak, Group Science, Engineering and Technology, E. Sabbelaan 53, B-8500 Kortrijk, Belgium

michielvdc@hotmail.com, Lieven.DeLathauwer@kuleuven.be.

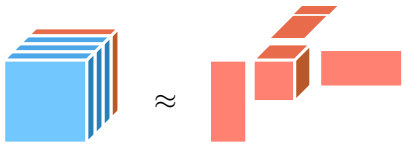


Fig. 1. Update of the LMLRA of a third-order tensor. The three factor matrices holding the subspaces are changed slightly and the third factor matrix is extended in the evolving direction. The core tensor changes as well.

extensive overview of the TMLUTVD, an analysis of its computational complexity and a comparison to the TMLSVD are given in [12].

In this article, we update the TMLUTVD in the scenario that is visualized in Figure 1. We also consider the tracking variant, where the third mode is observed through a sliding window: new slices are added at the last position in the third mode, old slices are removed at the first position, and the decomposition is updated accordingly. As the overall approximation problem is multi-*linear*, matrix tracking methods [7] can be applied to track the principal subspaces mode per mode, and afterwards, the estimates can be combined into a TMLUTVD [12]. To automatically determine the correct mode-n rank for all modes (the multilinear rank), we use a *rank-revealing* URV (RRURV) or ULV (RRULV) decomposition to track the principal subspaces instead of faster methods such as PAST/PASTd [13] and FAPI [14] that are not rank-revealing.

There exist alternative LMLRA updating methods that also track the mode-n principal subspaces through matrix subspace tracking methods. However, these methods do not exploit the multilinear structure of the problem to the same extent and/or do not allow for automatic determination of the multilinear rank. For instance, [15]–[17] track all subspaces simultaneously, but they neglect the multilinear structure and require the storage of large intermediate matrices. Using [18], the mode-n subspaces of a tensor can be tracked through small changes in the tensor, enabling streaming applications in relatively stationary environments. Next, in [19] and [20], sketching approaches are used to restrict the tensor to its principal subspace in each mode with the use of random projections following [21]. This allows the computation of an LMLRA with a single pass over all tensor entries, making the randomized methods suitable for streaming applications. Finally, [22] proposes a framework that can be combined with a range of matrix subspace tracking methods such as PAST and FAPI. As only the principal subspaces in every mode are stored between updates, automatic rank determination is not possible. Also, no sequential truncating strategies are employed when moving from one subspace to the next, although these can significantly speed up the algorithm in non-parallel settings.

Recapitulating, in this paper we will propose a new method for LMLRA tracking, which relies on the TMLUTVD. Contrary to existing methods, principal subspaces are determined with automatic rank estimation for each mode. It is even possible to upper-bound the approximation error of the TMLUTVD in every update [12]. On the other hand, TMLUTVD updating is less expensive than TMLSVD updating (as emphasized in [12], the MLUTVD can be computed exactly in a finite number of steps, while the MLSVD intrinsically requires an iterative procedure). In Section II, the basic TMLUTVD updating algorithm is discussed. Next, in Section III, explicit bounds are derived for the approximation error of the tracked TMLUTVD. In Section IV, three strategies are discussed that can significantly decrease the storage and computational requirements of the method. Finally, Section V shows some experiments on synthetic and real-life data. The algorithm is laid out for a real third-order tensor, updated in the third mode. Thus, mode 3 is hereafter considered as the evolving mode and its dimension changes during the updating process. The mode-1 and mode-2 subspaces are updated in every iteration as well, but as their dimensions remain the same, they are called non-evolving modes. Evidently, the method can also be applied to complex-valued tensors and any mode can be chosen as the updating mode. Extending the method to higher orders is straightforward, as this only increases the number of non-evolving modes.

A. Notation and terminology

Scalars, vectors and matrices are denoted by lowercase (a), bold lowercase ($\mathbf{a}$) and bold uppercase letters ($\mathbf{A}$), respectively, while tensors are written in calligraphic script ($\mathcal{A}$). The identity matrix and the zero matrix are denoted by $\mathbf{I}$ and $\mathbf{0}$, respectively. The Kronecker product of matrices is written as $\otimes$. In subscripts, a colon refers to all entries of a specific mode, e.g. $\mathbf{t}_{1:2}$ is a vector holding all entries of the third-order tensor $\mathcal{T}$ which have 1 and 2 as their first and third index, respectively. When formatting the entries of an Nth order tensor $\mathcal{T}$ in a matrix, the notation $\mathbf{T}_{[i_{m_1},i_{m_k};i_{m_{k+1}},i_{m_N}]}$ is used, with $[i_{m_1},i_{m_k};i_{m_{k+1}},i_{m_N}]$ providing the processing order of the different tensor modes in the row and column dimensions, i.e., the entries before the semicolon (;) give the mode processing order in the rows of the matrix, while those after the semicolon specify the order for the columns. If all modes appear in the column dimension, the semicolon is dropped and a lowercase letter is used for the resulting vector: for a tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, $\mathbf{t}_{[3,2,1]} \in \mathbb{R}^{I_1 I_2 I_3 \times 1}$ refers to the standard mode order for the vectorization of a third-order tensor. A mode-n fiber of a tensor is a vector obtained by fixing all indices but the nth. The mode-n unfoldings $\mathbf{T}_{[1;3,2]}$, $\mathbf{T}_{[2;3,1]}$ and $\mathbf{T}_{[3;2,1]}$ of a third order tensor $\mathcal{T}$ are constructed by stacking all its mode-n fibers as columns of a matrix, using the mode ordering specified by the notation. For example: in the mode-2 unfolding $\mathbf{T}_{[2;3,1]}$, the column ordering of the mode-2 fibers runs along mode 3 and mode 1, in that order, thus $t_{i_1 i_2 i_3}$ is mapped to $\left(\mathbf{T}_{[2;3,1]}\right)_{i_2,q}$, with $q = i_1 + (i_3 - 1)I_1$. The mode-n tensor-matrix product of a tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and a matrix $\mathbf{X} \in \mathbb{R}^{J \times I_n}$ is written as $\mathcal{T} \cdot_n \mathbf{X}$. For $n = 1$, 2 and 3, $\mathcal{T} \cdot_n \mathbf{X} \in \mathbb{R}^{J \times I_2 \times I_3}$, $\mathbb{R}^{I_1 \times J \times I_3}$ and $\mathbb{R}^{I_1 \times I_2 \times J}$, respectively. In unfolded form, computing the mode-n product is the same as computing $\mathbf{X}\mathbf{T}_{(n)}$. Kronecker products of the form $\mathbf{X}(\mathbf{Z} \otimes \mathbf{Y})$ are never explicitly formed, as they can be computed as matricized tensor times Kronecker products (MTKRONPRODs), for which specialized routines exist [23]. It suffices to store the matrices $\mathbf{Y}$ and $\mathbf{Z}$.

A low multilinear rank approximation (LMLRA) of a tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is defined as $\mathcal{T} \approx \mathcal{S} \cdot_1 \mathbf{U}_1 \cdot_2 \mathbf{U}_2 \cdot_3 \mathbf{U}_3$, where $\mathcal{S} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$, $\mathbf{U}_1 \in \mathbb{R}^{I_1 \times R_1}$, $\mathbf{U}_2 \in \mathbb{R}^{I_2 \times R_2}$, $\mathbf{U}_3 \in \mathbb{R}^{I_3 \times R_3}$. The vector (R_1, R_2, R_3) with $R_1 \leq I_1$, $R_2 \leq I_2$, $R_3 \leq I_3$ holds the number of (linearly independent) columns that are used in the factor matrices $\mathbf{U}_i$ of the approximation for every mode. If the approximation is exact and the $R^{(i)}$ are minimal, then this vector is called the multilinear rank of the tensor. One way to obtain the factor matrices of an LMLRA is to compute bases $\mathbf{U}_n$ for the principal column spaces of all mode-n unfoldings of the tensor. The core tensor $\mathcal{S}$ can then be found by forming $\mathcal{S} = \mathcal{T} \cdot_1 \mathbf{U}_1^{\mathrm{T}} \cdot_2 \mathbf{U}_2^{\mathrm{T}} \cdot_3 \mathbf{U}_3^{\mathrm{T}}$. While this construction method does not yield the least-squares optimal approximation, it is often only mildly suboptimal.

II. Multilinear UTV updating

Matrix tracking methods can be applied when a new signal vector becomes available at certain time intervals. Consider the matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$, where $\mathbf{x}_{:j}$ represents a sample obtained at time j. The SVD $\mathbf{X} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$ of $\mathbf{X}$ consists of matrices $\mathbf{U} \in \mathbb{R}^{I \times R}$, $\mathbf{\Sigma} \in \mathbb{R}^{R \times R}$ and $\mathbf{V} \in \mathbb{R}^{J \times R}$, with $R \leq I$, $\mathbf{U}$ and $\mathbf{V}$ column-wise orthonormal and $\mathbf{\Sigma}$ diagonal. A time sample $\mathbf{x}_{:j}$ is represented as $\mathbf{U}\mathbf{\Sigma}\mathbf{v}_{j:}^{\mathrm{T}}$, so the green column of $\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$ in Figure 2 (top) contains the coefficients with which the green column of $\mathbf{X}$ can be expressed in the basis formed by the columns of $\mathbf{U}$. Conversely, $\mathbf{U}^{\mathrm{T}}\mathbf{x}_{:j} \in \mathbb{R}^R$ is a low dimensional representation of $\mathbf{x}_{:j} \in \mathbb{R}^I$. When new samples arrive as extra columns of $\mathbf{X}$, $\mathbf{U}$ should be updated so that it represents the principal column space of the updated $\mathbf{X}$. The new $\mathbf{U}$ then allows us to find low dimensional approximations of any time sample in $\mathbf{X}$.

If each time sample is not a vector, but a matrix or even a tensor instead, one obtains a tensor tracking problem. More meaningful approximations can often be found by keeping the tensor structure intact. Let $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ have frontal slices $\mathbf{T}_{::k}$ representing the sample obtained at time k. Then, an LMLRA $\mathcal{C} \cdot_1 \mathbf{U}_1 \cdot_2 \mathbf{U}_2 \cdot_3 \mathbf{U}_3$ of $\mathcal{T}$ can be obtained by extracting the principal subspaces of $\mathcal{T}$ in all three modes [5]. Bases for these subspaces are collected in the factor matrices $\mathbf{U}_n \in \mathbb{R}^{I_n \times R_n}$, for $n = 1, 2, 3$. The core tensor $\mathcal{C} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ holds the interactions between modes. Unfolded, the approximation has the following form: $\mathbf{T}_{[2,1;3]} = (\mathbf{U}_2 \otimes \mathbf{U}_1)\mathbf{C}_{[2,1;3]}\mathbf{U}_3^{\mathrm{T}}$, see Figure 2 (bottom). In the tensor case, there is a similar relationship as in the matrix case between the green column of $\mathbf{C}_{[2,1;3]}\mathbf{U}_3^{\mathrm{T}}$, the green column of $\mathbf{T}_{[2,1;3]}$ and the columns of $\mathbf{U}_2 \otimes \mathbf{U}_1$. However, a stronger interpretation is possible. Namely, for all k, we have that $\mathbf{T}_{::k} = \mathbf{U}_1\mathbf{Y}_k\mathbf{U}_2^{\mathrm{T}}$, with $\mathbf{T}_{::k} \in \mathbb{R}^{I_1 \times I_2}$ and with $\mathbf{Y}_k \in \mathbb{R}^{R_1 \times R_2}$ the matricization of the kth column of $\mathbf{C}_{[2,1;3]}\mathbf{U}_3^{\mathrm{T}}$. In other words, $\mathbf{U}_1$ and $\mathbf{U}_2$ admit a *low-rank* representation of any frontal slice of $\mathcal{T}$.

When new samples need to be included in the tensor or old samples need to be discarded, stable and efficient updates/downdates should be applied to the mode-1 and mode-2 principal subspaces. The case where only the subspaces of the non-evolving modes 1 and 2 are tracked is outlined in the conference paper [9]. In the more general case proposed below, the principal subspace of the evolving mode 3 is tracked as well. If the full LMLRA of the tensor is required, the core tensor can also be tracked.

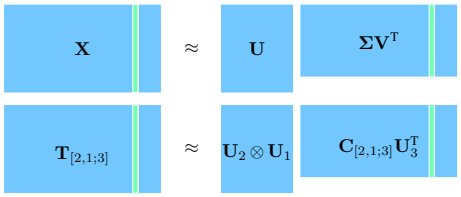


Fig. 2. Top: low-rank approximation of a matrix $\mathbf{X}$ as the product of the matrices $\mathbf{U}$ and $\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$. One time-sample is highlighted in $\mathbf{X}$ and $\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$. Bottom: Third-order extension of the factorization above. The samples are frontal slices of a tensor $\mathcal{T}$ and the LMLRA is given by $\mathbf{T}_{[2,1;3]} = (\mathbf{U}_2 \otimes \mathbf{U}_1)\mathbf{C}_{[2,1;3]}\mathbf{U}_3^{\mathrm{T}}$. As for the matrix case, one time sample is highlighted in $\mathbf{T}_{[2,1;3]}$ and $\mathbf{C}_{[2,1;3]}\mathbf{U}_3^{\mathrm{T}}$.

When $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is tracked with a sliding window of length w, new slices are added and removed in the updating mode. Only after adding w slices, do we start to remove the oldest slice. From then on, the tracked tensor is always of dimensions $(I_1 \times I_2 \times w)$. When no slices are removed, the window is said to be infinite and the tracked tensor at update n has dimensions $(I_1 \times I_2 \times n)$. Superscripts of objects refer to the index of the update in which they have been computed, so $\mathbf{X}^{(n)}$ refers to a matrix computed in the nth updating step.

Computing the LMLRA that provides the optimal approximation for the given rank is often impractically costly, especially in an updating context. A much cheaper alternative, which is often very close to optimal, is the truncated MLSVD (TMLSVD). Starting from the MLSVD [5], the TMLSVD is obtained by truncating in every mode the singular vectors that do not lie in the principal subspace and the corresponding parts of the core, see Figure 3. To obtain the TMLSVD of an Nth order tensor, N SVDs of unfolded tensors need to be computed, which can still require considerable computational effort. In many practical applications, however, it is not necessary to compute all information that the TMLSVD offers. It can be sufficient to split each mode-n subspace into its principal subspace, say of dimension R_n, and the orthogonal complement thereof. Ideally, the mode-n principal subspace corresponds to the underlying signal, while the orthogonal complement corresponds to noise. For each mode, this split can be performed using a decomposition that is cheaper than the SVD and it is not necessary to explicitly compute all multilinear singular values. One can even use a decomposition that can be computed in a finite number of steps instead of the SVD whose computation is inherently an iterative process. Using a rank-revealing UTV decomposition (either a ULV or URV decomposition), for instance, one obtains a multilinear UTV decomposition (MLUTVD). See [12] for details, including bounds on the approximation error. Like the MLSVD, the MLUTVD allows for a truncation to an LMLRA, see Figure 3, with a truncation error that is close to minimal.

Below, we supply the basic steps to perform a TMLUTVD update. Later, we extend this basic algorithm to make it more efficient, both computationally and with respect to storage requirements, further facilitating its utilization in real-time

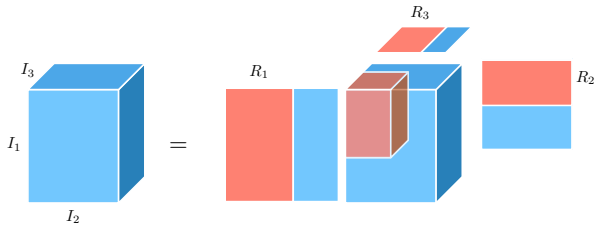


Fig. 3. Both the MLSVD and the MLUTVD can be truncated to obtain an LMLRA of a tensor by only using the columns of the factor matrices that correspond to the principal mode-n subspaces and the corresponding entries of the core tensor, marked in red.

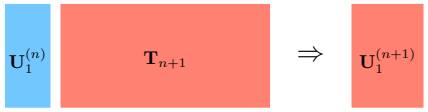


Fig. 4. Update of the LMLRA for mode 1. The new columns of the mode-1 unfolding are included in the subspace $\mathbf{U}_1^{(n)}$.

applications. To perform an update of the TMLUTVD, one needs to update the principal column spaces of the unfoldings of $\mathcal{T}$ in the non-evolving modes 1 and 2, and the subspace in the evolving mode 3. Section II-A covers the non-evolving modes and can be seen as a two-way block-update extension of the matrix technique in [24]. We explain the rationale, briefly summarize the matrix approach, and tailor the method to our tensor setting. Section II-B concerns the update of the subspace in the evolving mode, which is quite different from the update of the non-evolving modes. An indication of this is given in Figure 2 (bottom): the matrix $\mathbf{C}_{[2,1;3]}\mathbf{U}_3^{\mathrm{T}}$ to the right represents a (low) rank-R_3 factorization of an $(R_1R_2 \times K)$ matrix. This specific aspect of low-rank factorization does not manifest itself in the matrix case in Figure 2 (top), where just $\mathbf{\Sigma} \in \mathbb{R}^{R\times R}$ and $\mathbf{V}^{\mathrm{T}} \in \mathbb{R}^{R\times K}$. Further, in Section II-C, the storage and computational costs of the algorithm are compared to those of the batch method and the choice of adequate truncation threshold values is discussed in Section II-D.

A. Subspace tracking of the non-evolving modes

For mode 1, of which the dimension of the ambient space (I_1) is constant, the goal is to track the column space of the mode-1 unfolding $\mathbf{T}_{[1;3,2]}$. Assume that this space has dimension $R_1^{(n)}$ after n slices have been added, with $1 \leq R_1^{(n)} \leq I_1$. When a new slice $\mathbf{T}_{n+1} := \mathbf{T}_{::n+1} \in \mathbb{R}^{I_1\times I_2}$ is added to the tensor, this corresponds to adding all mode-1 fibers of the slice as columns to $\mathbf{T}_{[1;3,2]}^{(n)}$, see Figure 4. Thus, the mode-1 subspace should take every mode-1 fiber of $\mathbf{T}_{n+1}$ into account. Likewise, when an old slice is removed, the contributions of all mode-1 fibers of that old slice should be removed.

We first set a threshold γ_1 which is used to decide whether a singular vector of a matrix $\mathbf{X}$ belongs to its principal subspace or to the orthogonal complement. The singular values of $\mathbf{X}$ are the square roots of the eigenvalues $\lambda_i \geq 0$ of $\mathbf{X}\mathbf{X}^{\mathrm{T}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$, with $\mathbf{U}$ orthogonal and the diagonal matrix $\mathbf{\Lambda}$ holding the sorted eigenvalues. The threshold γ_1 is chosen such that the large eigenvalues $\lambda_i \geq \gamma_1^2$ correspond to the underlying signal, while small eigenvalues $\lambda_i < \gamma_1^2$ correspond to noise. Ideally, γ_1 is chosen within a clear gap between the large and small eigenvalues. The k columns of $\mathbf{U}$ that belong to the retained eigenvalues then span the principal subspace of $\mathbf{X}$. Consequently, singular vectors corresponding to singular values above γ_1 are considered to be part of the principal subspace and the other singular values as corresponding to noise. In Section II-D, strategies are discussed for setting γ_1.

As explained in Section I, we only wish to track the principal subspace of $\mathbf{T}_{[1;3,2]}$ and not the individual singular vectors, so any principal subspace-revealing decomposition can be updated instead of the SVD. Here, a signed RRULV decomposition (RRULVD) is used, which is less expensive to compute than an SVD. While its resulting subspace is not the same as the one obtained with the SVD, it typically lies quite close. When computing an RRULVD, one can exploit the fact that if $\mathbf{X}$ has d singular values above γ_1, then $\mathbf{X}\mathbf{X}^{\mathrm{T}} - \mathbf{N}\mathbf{N}^{\mathrm{T}}$ has d eigenvalues above zero for any $\mathbf{N}$ such that $\mathbb{E}(\mathbf{N}\mathbf{N}^{\mathrm{T}}) = \gamma_1^2\mathbf{I}$. By separating the subspace corresponding to the positive eigenvalues from the one corresponding to the negative eigenvalues, the rank of the principal subspace with respect to the threshold γ_1 can be extracted exactly and a good estimate of the principal subspace can be obtained.

After processing n tensor slices, the signed RRULVD of the mode-1 unfolding of the tensor has the following form [24]:

$$\mathbf{Q}_1^{(n)^{\mathrm{T}}}[\mathbf{N}_1^{(n)}, \mathbf{S}_1^{(n)}]\mathbf{\Phi}_1^{(n)} = [\mathbf{L}_N^{(n)}, \mathbf{0}, \mathbf{L}_S^{(n)}, \mathbf{0}]. \tag{1}$$

Herein, $\mathbf{S}_1^{(n)} \in \mathbb{R}^{I_1\times I_2 w}$ is a matrix holding all mode-1 fibers that are currently within the sliding window, while $\mathbf{N}_1^{(n)}\mathbb{R}^{I_1\times I_2(n-w)}$ is a matrix holding all mode-1 fibers that have been dropped from the sliding window and are thus no longer included in the subspace computation. Thus $[\mathbf{N}_1^{(n)}, \mathbf{S}_1^{(n)}]$ is a matrix representation of all processed mode-1 fibers that grows as more updates are performed. The matrix $\mathbf{Q}_1^{(n)} \in \mathbb{R}^{I_1\times I_1}$ is orthogonal, $\mathbf{\Phi}_1^{(n)} \in \mathbb{R}^{I_2 n\times I_2 n}$ is J-orthonormal and $\mathbf{L}_1^{(n)} = [\mathbf{L}_N^{(n)}, \mathbf{L}_S^{(n)}] \in \mathbb{R}^{I_1\times I_1}$ is lower triangular. The matrix $\mathbf{\Phi}_1^{(n)}$ being J-orthonormal means that $\mathbf{\Phi}_1^{(n)}\mathbf{J}\mathbf{\Phi}_1^{(n)^{\mathrm{T}}} = \mathbf{J}$ and $\mathbf{\Phi}_1^{(n)^{\mathrm{T}}}\mathbf{J}\mathbf{\Phi}_1^{(n)} = \mathbf{J}$ hold for $\mathbf{J} = \mathrm{diag}([1; \ldots; 1; -1; \ldots; -1])$. The positive-negative structure of $\mathbf{J}$ follows the block-partitioning of $\mathbf{\Phi}_1^{(n)}$, which in turn depends on γ_1. Because of the J-orthonormal structure of $\mathbf{\Phi}_1^{(n)}$, columns of $\mathbf{L}_S^{(n)} \in \mathbb{R}^{I_1\times R_1^{(n)}}$, which are the columns of $\mathbf{L}_1^{(n)}$ corresponding to the principal mode-n subspace, have a different signature than columns that correspond to the orthogonal complement, stored in $\mathbf{L}_N^{(n)}$. Consequently, the principal subspace can be obtained by splitting the columns of $\mathbf{Q}_1^{(n)}$ into orthonormal bases for the principal column space $\mathbf{Q}_S^{(n)} \in \mathbb{R}^{I_1\times R_1^{(n)}}$ and for its orthogonal complement $\mathbf{Q}_N^{(n)} \in \mathbb{R}^{I_1\times(I_1-R_1^{(n)})}$: $\mathbf{Q}_1^{(n)} = [\mathbf{Q}_N^{(n)}|\mathbf{Q}_S^{(n)}]$. The signed RRULVD can be updated in a stable and non-iterative way using the Signed URV (SURV) method [24] with a complexity comparable to that of a QR update ($\mathcal{O}(I_1^2)$ flop). Both exponential and sliding windows can be handled in the SURV algorithm. Compared to other updating methods, the latter is particularly easy, as updates and downdates of new columns are handled similarly. Below, we consider sliding windows.

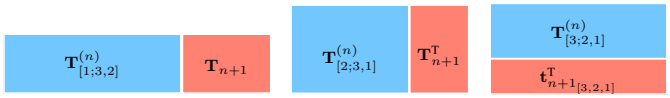


Fig. 5. Appending a new slice to the tensor in the evolving mode 3 (right) corresponds to adding a new row to the mode-3 unfolding, while for modes 1 (left) and 2 (right), the new slice is added in the column dimension.

If a new slice $\mathbf{T}_{n+1}$ is added to the tensor, its columns are appended to $\mathbf{S}_1^{(n)}$. The decomposition (1) is thus updated to include the new slice, which results in a new decomposition

$$\mathbf{Q}_1^{(n+1)^{\mathrm{T}}}[\mathbf{N}_1^{(n+1)}, \mathbf{S}_1^{(n+1)}]\mathbf{\Phi}_1^{(n+1)} = [\mathbf{L}_N^{(n+1)}, \mathbf{0}, \mathbf{L}_S^{(n+1)}, \mathbf{0}],$$

with $\mathbf{N}_1^{(n+1)} = \mathbf{N}_1^{(n)}$ and $\mathbf{S}_1^{(n+1)} = [\mathbf{S}_1^{(n)}, \mathbf{T}_{n+1}]$. The matrix $\mathbf{S}_1^{(n)} \in \mathbb{R}^{I_1 \times I_2 w}$ and the possibly very large matrices $\mathbf{N}_1^{(n)} \in \mathbb{R}^{I_1 \times I_2 (n-w)}$ and $\mathbf{\Phi}_1^{(n)} \in \mathbb{R}^{I_2 n \times I_2 n}$, which grow with every slice update, are never stored during the updating process, making the algorithm memory-efficient as well. The algorithm is initialized with $\mathbf{L}_1^{(0)} = \gamma_1 \mathbf{I}$, $\mathbf{Q}_1^{(0)} = \mathbf{I}$, $\mathbf{N}_1^{(0)}$ empty and $\mathbf{S}_1^{(0)} = \gamma_1 \mathbf{I}$ to implicitly set γ_1 as splitting threshold.

The SURV updating algorithm applies hyperbolic rotations and Givens row and column rotations (see the overview in Table I of [24]) to maintain the lower-triangular structure of $\mathbf{L}_1^{(n)}$, while being careful that only stable hyperbolic rotations are used [24]. The matrix $\mathbf{L}_1^{(n)}$ holds the weights of the columns in $[\mathbf{Q}_N^{(n)} | \mathbf{Q}_S^{(n)}]$ and is tracked alongside $\mathbf{Q}_1^{(n)}$. Using $\mathbf{L}_1^{(n)}$, the dimension of the subspace is automatically adjusted: subspace vectors that contribute to the subspace by more than the predefined threshold value γ_1 are moved to $\mathbf{Q}_S^{(n+1)} \in \mathbb{R}^{I_1 \times R_1^{(n+1)}}$, while the other vectors are kept in $\mathbf{Q}_N^{(n+1)} \in \mathbb{R}^{I_1 \times (I_1 - R_1^{(n+1)})}$ [24]. The matrix representation $\mathbf{Q}_S^{(n+1)}$ of the principal mode-1 subspace then takes the role of the mode-1 factor matrix $\mathbf{U}_1^{(n+1)}$ in the new LMLRA.

For mode 2, of which the dimension of the ambient space (I_2) does not evolve either, the same approach is used to update the mode-2 subspace up to a threshold value of γ_2 with all columns of the transposed new slice $\mathbf{T}_{n+1}^{\mathrm{T}} \in \mathbb{R}^{I_2 \times I_1}$.

When a sliding window is used, both slice updates and downdates are performed to keep the window length constant. For the non-evolving modes 1 and 2, downdating an old slice $\mathbf{T}_{::n-w}$ is very similar to updating a new slice. The columns of the old slice are appended to $\mathbf{N}_1^{(n)}$ instead of $\mathbf{S}_1^{(n)}$ and the decomposition is updated with SURV downdates [24], whereby one SURV downdate is performed for every column of $\mathbf{T}_{::n-w}$.

In Algorithm 1, the update of the non-evolving modes is shown in pseudocode. The functions `SURV_up` and `SURV_down` refer to the SURV updating and SURV downdating steps described in [24], respectively. The pseudocode may help the reader to keep track of (the indices in) the notation as the updates are constructed.

Algorithm 1: SURV updating of the non-evolving modes 1 and 2

```
function [Q_i^(n+1), L_i^(n+1), R_i^(n+1)] =
  SURV_nonevol(Q_i^(n), L_i^(n), R_i^(n), γ_i, T_new, T_old)
Input : SURV matrices Q_i^(n), L_i^(n), rank values R_i^(n)
        and thresholds γ_i for i = 1, 2, new slice
        T_new, old slice T_old
Output: SURV matrices Q_i^(n+1), L_i^(n+1) and rank
        values R_i^(n+1) for i = 1, 2
for i = 1, 2 do
    % SURV update and downdate for mode-1 and 2
    [Q_i^(n+1), L_i^(n+1), R_i^(n+1)] ←
      SURV_up(Q_i^(n), L_i^(n), R_i^(n), γ_i, T_new)
    [Q_i^(n+1), L_i^(n+1), R_i^(n+1)] ←
      SURV_down(Q_i^(n+1), L_i^(n+1), R_i^(n+1), γ_i, T_old)
    % Extract factor matrices from Q_i-matrices
    U_i^(n+1) ← Q_i^(n+1)_{:, I_1 − R_i^(n+1)+1:I_i}
end
```

B. Subspace tracking of the evolving mode

For the non-evolving modes 1 and 2, adding a new slice corresponds to the inclusion of new columns into the mode-n unfolding. As we are only concerned about the principal subspace, the column order does not matter. Consequently, both non-evolving modes can be handled with the SURV updating method. For the mode-3 unfolding, however, the new slice takes the form of one long vector $\mathbf{t}^{\mathrm{T}}$, appended to the bottom of $\mathbf{T}^{(n)}_{[3;2,1]}$, see Figure 5. A different approach is thus required for the evolving mode 3.

In many applications, one is mainly interested in tracking the principal row space of $\mathbf{T}^{(n)}_{[3;2,1]}$ of dimension R_3, assumed to be stationary or slowly-evolving [22]. For instance, consider the following 2D harmonic retrieval application. Two far-field signals are observed on a rectangular array, with one signal arriving via two different paths and the other via a single path. A suitable LMLRA $\mathcal{T} \approx \mathcal{C} \cdot_1 \mathbf{U}_1 \cdot_2 \mathbf{U}_2 \cdot_3 \mathbf{U}_3$ would have $R_1 = R_2 = 3$ and $R_3 = 2$, with the first two modes corresponding to the spatial dimensions and the third mode to the time dimension. The subspaces $\mathrm{col}(\mathbf{U}_1)$ and $\mathrm{col}(\mathbf{U}_2)$ give the azimuth and elevation of the directions of arrival (DOAs), $\mathrm{col}(\mathbf{C}_{[2,1;3]})$ describes how the paths and transmitted signals are connected and $\mathbf{U}_3$ corresponds to the transmitted signal values. Only $\mathrm{col}(\mathbf{U}_1)$, $\mathrm{col}(\mathbf{U}_2)$ and $\mathrm{col}(\mathbf{C}_{[2,1;3]})$ are slowly-varying and can thus be tracked. The transmitted signals at any time instance can be recovered up to rotation through the computation of the corresponding row of $\mathbf{U}_3$.

Figure 5 shows that the matrix $\mathbf{T}^{(n)}_{[3;2,1]}$ for which we want to track the stationary row space obtains a new row in every update. Consequently, if we transpose the problem, we track the stationary column space of a matrix that receives a new column in every update. Note that this is precisely the same setting as for modes 1 and 2. We can thus transpose the problem and apply the SURV approach [24] as before to track the principal column space of $\mathbf{T}^{(n)}_{[2,1;3]} = \mathbf{T}^{(n)^{\mathrm{T}}}_{[3;2,1]}$, which corresponds to the principal row space of $\mathbf{T}^{(n)}_{[3;2,1]}$. The core tensor $\mathcal{C}^{(n)} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ can then easily be computed through $\mathbf{C}^{(n)}_{[2,1;3]} = (\mathbf{U}_2^{(n)^{\mathrm{T}}} \otimes \mathbf{U}_1^{(n)^{\mathrm{T}}})\mathbf{Q}_S\mathbf{L}_{22}$, with $\mathbf{Q}_S$ and

$\mathbf{L}_{22}$ originating from the SURV decomposition of $\mathbf{T}^{(n)}_{[2,1;3]}$ and $\mathbf{L}_{22}$ referring to the bottom right $(R_3 \times R_3)$-block of $\mathbf{L}$ [24]. Together, $\mathbf{U}^{(n)}_1$, $\mathbf{U}^{(n)}_2$ and $\mathcal{C}^{(n)}$ describe the stationary subspace that is tracked: the matrix $(\mathbf{U}^{(n)}_2 \otimes \mathbf{U}^{(n)}_1)\mathbf{C}^{(n)}_{[2,1;3]}$ takes on a similar function as the matrix $\mathbf{U}$ in matrix subspace tracking methods, where one tracks a decomposition of the form $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$, as explained in the beginning of this section. Similarly, the factor matrix $\mathbf{U}^{(n)}_3$ of the LMLRA corresponds to the $\mathbf{V}$-matrix of the matrix methods. It is often unnecessary to explicitly form this matrix: if required, a row of $\mathbf{U}^{(n)}_3$ can be obtained by expressing the corresponding column of $\mathbf{T}^{(n)}_{[2,1;3]}$ in the basis generated by $(\mathbf{U}^{(n)}_2 \otimes \mathbf{U}^{(n)}_1)\mathbf{C}^{(n)}_{[2,1;3]}$.

A drawback of the SURV approach described above is that it is not particularly fast. The vectorized new slices have $I_1 I_2$ entries which makes updates and downdates computationally costly (one update has a cost of $\mathcal{O}(I_1^2 I_2^2)$ flop, see Section II-C2). A priori, there are two ways to decrease the computational cost. First, one can exploit the low multilinear rank to work with matrices with $R_1 R_2$ instead of $I_1 I_2$ rows and/or columns through the use of sequential truncation. Second, one can reduce the dimension from $I_1 I_2$ to w when $w < I_1 I_2$, as the mode-3 rank can never exceed the window length. Neither strategy is easy to combine with the SURV approach, though, because of the fact that the $\mathbf{\Phi}$-matrix is never stored [24]. Let us make this more intuitive, without going into technical details. For the case of sequential truncation, the reduction from I_1 to R_1 brings no difficulties for mode 2: it simply reduces the number of columns to be appended. However, there is a deeper effect in mode-3, as the dimension of the ambient space changes from $I_1 I_2$ to $R_1 R_2$; this makes it more difficult to assess the accumulated error and, hence, hinders future decisions about increasing or decreasing the mode-3 rank or error bounds. In the rest of this text, we therefore choose to track a two-sided decomposition instead of the SURV decomposition. In Section IV, we show that updates and downdates of this two-sided decomposition can be sped up very significantly by exploiting small window lengths ($w < I_1 I_2$), compressing the new slice and applying sequential truncation, in contrast to the SURV updates and downdates. Additionally, the two-sided decomposition gives us provable accuracy bounds for the approximation of $\mathcal{T}$ that it provides, see Section III. Again, as we only need to determine the principal subspaces up to a certain noise threshold γ_3 and not the individual singular vectors, we use an RRURV decomposition (RRURVD) [10] to estimate the principal column and row spaces, as this is more efficient computationally than tracking an SVD. The remainder of this section discusses the basic method without compression and with $w \geq I_1 I_2$: we show how the RRURVD of $\mathbf{T}^{(n)}_{[3;2,1]}$ can be updated and downdated for the addition and removal of rows, without yet making attempts to reduce the storage or computational costs, which is addressed in Section IV.

Denote by $\mathbf{Q}^{(n)}_3 \mathbf{R}^{(n)}_3 \mathbf{V}^{(n)^{\mathrm{T}}}_3$ the RRURVD of $\mathbf{T}^{(n)}_{[3;2,1]}$ after update n, thus $\mathbf{Q}^{(n)}_3 \in \mathbb{R}^{w \times I_1 I_2}$, is column-wise orthonormal, $\mathbf{R}^{(n)}_3 \in \mathbb{R}^{I_1 I_2 \times I_1 I_2}$ is upper triangular and $\mathbf{V}^{(n)}_3 \in \mathbb{R}^{I_1 I_2 \times I_1 I_2}$ is column-wise orthonormal. Using the mode-3 threshold value γ_3, the $\mathbf{R}$-matrix of the RRURVD has the following form [10]:

$$\mathbf{R}^{(n)}_3 = \begin{bmatrix} \mathbf{E}^{(n)}_3 & \mathbf{F}^{(n)}_3 \\ \mathbf{0} & \mathbf{G}^{(n)}_3 \end{bmatrix},$$

with $\|\mathbf{F}^{(n)}_3\|^2 + \|\mathbf{G}^{(n)}_3\|^2 \leq \gamma_3^2$ and $\mathbf{E}^{(n)}_3 \in \mathbb{R}^{R^{(n)}_3 \times R^{(n)}_3}$ and $\mathbf{G}^{(n)}_3$ upper triangular. We can obtain the principal column space of $\mathbf{T}^{(n)}_{[3;2,1]}$ by taking the span of the R_3 leftmost columns of $\mathbf{Q}^{(n)}_3$, as the other columns will not contribute more than γ_3 to the norm of $\mathbf{T}^{(n)}_{[3;2,1]}$ [10]. Analogously, we can obtain the principal row space as the subspace spanned by the R_3 leftmost columns of $\mathbf{V}^{(n)}_3$. Additionally, the mode-3 unfolding of the core tensor of the TMLUTVD can be obtained by computing $\mathbf{C}^{(n)}_{[3;2,1]} = \mathbf{R}^{(n)}_{3_{1:R_3},1:R_3} \mathbf{V}^{(n)^{\mathrm{T}}}_{3_{:,1:R_3}} (\mathbf{U}^{(n)}_2 \otimes \mathbf{U}^{(n)}_1) \in \mathbb{R}^{R_3 \times R_1 R_2}$. Below, we give an approach to find the updated decomposition $\mathbf{Q}^{(n+1)}_3 \mathbf{R}^{(n+1)}_3 \mathbf{V}^{(n+1)^{\mathrm{T}}}_3$ after adding a new slice.

To include the information of the new row $\mathbf{t}^{\mathrm{T}}$ in the old decomposition $\mathbf{Q}^{(n)}_3 \mathbf{R}^{(n)}_3 \mathbf{V}^{(n)^{\mathrm{T}}}_3$, we can write

$$\begin{bmatrix} \mathbf{T}^{(n)}_{[3;2,1]} \\ \mathbf{t}^{\mathrm{T}} \end{bmatrix} \approx \begin{bmatrix} \mathbf{Q}^{(n)}_3 & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}^{(n)}_3 \\ \mathbf{x}^{\mathrm{T}} \end{bmatrix} \mathbf{V}^{(n)^{\mathrm{T}}}_3 = \begin{bmatrix} \mathbf{Q}^{(n)}_3 \mathbf{R}^{(n)}_3 \mathbf{V}^{(n)^{\mathrm{T}}}_3 \\ \mathbf{x}^{\mathrm{T}} \mathbf{V}^{(n)^{\mathrm{T}}}_3 \end{bmatrix}. \quad (2)$$

In (2), the vector $\mathbf{x}^{\mathrm{T}} \mathbf{V}^{(n)^{\mathrm{T}}}_3 \in \mathbb{R}^{I_1 I_2}$ is exactly equal to the new row $\mathbf{t}^{\mathrm{T}}$. The new $\mathbf{R}$-matrix can be brought back to upper triangular form by zeroing the entries of $\mathbf{x}^{\mathrm{T}}$ one by one with the use of Givens row and column rotations without destroying the rank-revealing structure, following the scheme from [25]. For the rightmost $(I_1 I_2 - R_3 - 1)$ entries of the bottom row of the $\mathbf{R}$-matrix, Givens column rotations are used to zero the entries from right to left, with each Givens column rotation followed by a Givens row rotation to erase the resulting fill-in. Afterwards, the leftmost (R_3+1) entries of the bottom row are zeroed from left to right using Givens row rotations. All row and column rotations should be absorbed in the new $\mathbf{Q}$- and $\mathbf{V}$-matrices, respectively, to obtain a new URV decomposition $\mathbf{Q}_{\text{up}} \mathbf{R}_{\text{up}} \mathbf{V}^{\mathrm{T}}_{\text{up}}$ of $\mathbf{T}^{(n+1)}_{[3;2,1]}$, wherein $\mathbf{Q}_{\text{up}} \in \mathbb{R}^{(w+1) \times (I_1 I_2 + 1)}$, $\mathbf{R}_{\text{up}} \in \mathbb{R}^{(I_1 I_2 + 1) \times I_1 I_2}$ and $\mathbf{V}_{\text{up}} \in \mathbb{R}^{I_1 I_2 \times I_1 I_2}$.

After the update, a downdate of the mode-3 subspace can be performed with QR-downdates [10], [11]. Starting from the updated URV decomposition $\mathbf{Q}_{\text{up}} \mathbf{R}_{\text{up}} \mathbf{V}^{\mathrm{T}}_{\text{up}}$, we apply elementary rotations to zero the first row of $\mathbf{Q}_{\text{up}}$. Applying the inverse rotations to $\mathbf{R}_{\text{up}}$ then gives us the decomposition

$$\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{\text{dn}} \end{bmatrix} \begin{bmatrix} \mathbf{y}^{\mathrm{T}} \\ \mathbf{R}_{\text{dn}} \end{bmatrix} \mathbf{V}^{\mathrm{T}}_{\text{up}}.$$

The matrix $\mathbf{V}_{\text{up}}$ is unchanged, so by setting $\mathbf{V}_{\text{dn}} = \mathbf{V}_{\text{up}}$, $\mathbf{Q}_{\text{dn}} \mathbf{R}_{\text{dn}} \mathbf{V}^{\mathrm{T}}_{\text{dn}}$ is a URV decomposition of $\mathbf{T}^{(n+1)}_{[3;2,1]}$ after downdating its first row.

Both the update and the downdate preserve the URV-form of the decomposition and the small values in the rightmost columns of $\mathbf{R}^{(n)}_3$. Still, an increase or decrease by one of the rank of the $\mathbf{R}$-matrix might be in order (adding or removing a row can only increase or decrease the rank by one, respectively). The non-rank-revealing URV decomposition $\mathbf{Q}_{\text{dn}} \mathbf{R}_{\text{dn}} \mathbf{V}^{\mathrm{T}}_{\text{dn}}$ with $\mathbf{R}_{\text{dn}} \in \mathbb{R}^{I_1 I_2 \times I_1 I_2}$ can be brought back into RRURV form with the method from [25]. This method

uses an iterative approach, wherein the matrix $\mathbf{R}_{\text{dn}}$ is deflated until the requested rank-revealing upper triangular form is obtained. The required Givens rotations for the deflation are absorbed into the matrices $\mathbf{U}_{\text{dn}}$ and $\mathbf{V}_{\text{dn}}$ to obtain the downdated URV decomposition $\mathbf{Q}_3^{(n+1)}\mathbf{R}_3^{(n+1)}{\mathbf{V}_3^{(n+1)}}^{\mathrm{T}}$. In our case, at most one deflation step is required to restore the rank-revealing property.

A high-level overview of the procedure for the evolving mode 3 is given as Algorithm 2. The functions `URV_up`, `QR_down` and `deflate` refer to the URV update, the QR downdate and the deflation steps, respectively, that have been outlined in this subsection.

Algorithm 2: RRURV tracking for the evolving mode

function $[\mathbf{Q}_3^{(n+1)},\mathbf{R}_3^{(n+1)},\mathbf{V}_3^{(n+1)},\mathbf{U}_3^{(n+1)},R_3^{(n+1)}]$ = RRURV_evol($\mathbf{Q}_3^{(n)}$, $\mathbf{R}^{(n)}$, $\mathbf{V}_3^{(n}$, $R_3^{(n)}$ γ_3, $\mathbf{t}$)

Input : RRURV matrices $\mathbf{Q}_3^{(n)}$, $\mathbf{R}_3^{(n)}$, $\mathbf{V}_3^{(n)}$, rank $R_3^{(n)}$, threshold γ_3, vectorized new slice $\mathbf{t}$

Output: RRURV matrices $\mathbf{Q}_3^{(n+1)}$, $\mathbf{R}_3^{(n+1)}$, $\mathbf{V}_3^{(n+1)}$, rank $R_3^{(n)}$, mode-3 factor matrix $\mathbf{U}_3^{(n+1)}$

% RRURV update and QR downdate

$[\mathbf{Q}_{\text{up}}, \mathbf{R}_{\text{up}}, \mathbf{V}_{\text{up}}] \leftarrow \texttt{URV_up}(\mathbf{Q}_3^{(n)}, \mathbf{R}_3^{(n)}, \mathbf{V}_3^{(n)}, \mathbf{t})$

$[\mathbf{Q}_{\text{dn}}, \mathbf{R}_{\text{dn}}] \leftarrow \texttt{QR_down}(\mathbf{Q}_{\text{up}}, \mathbf{R}_{\text{up}})$

% Restore rank-revealing structure and extract $\mathbf{U}_3^{(n+1)}$

$[\mathbf{Q}_3^{(n+1)}, \mathbf{R}_3^{(n+1)}, \mathbf{V}_3^{(n+1)}, R_3^{(n+1)}] \leftarrow$ $\texttt{deflate}(\mathbf{Q}_{\text{dn}}, \mathbf{R}_{\text{dn}}, \mathbf{V}_{\text{up}}, R_3^{(n)}, \gamma_3)$

$\mathbf{U}_3^{(n+1)} \leftarrow \mathbf{Q}_{3_{:,1:R_3}}^{(n+1)}$

C. Algorithm complexity

Ideally, an updating method has both a lower computational and storage cost than its corresponding batch algorithm, as computation time and storage space are typically limited in tracking applications. Below, we show that while the basic algorithm from Sections II-A and II-B has a provably upper bounded approximation error, its memory and computational complexity do not yet offer significant reductions compared to the batch algorithm. Therefore, in Section IV, further optimizations are discussed to dramatically reduce both costs.

1) Storage cost: The tensor has $I_1 I_2 w$ entries. For large values of I_1, I_2 and w, storing the windowed tensor can become costly. For finite window lengths, the proposed updating method essentially still requires all slices within the sliding window to be stored, as these slices are used to downdate the subspaces in later iterations. As a slice is only downdated after w updates, though, it can be stored in non-local memory. All $I_1 I_2$ entries of the new slice have to be stored as well.

For the SURV updating of a non-evolving mode, say mode 1, only the matrices $\mathbf{Q}_1^{(n)} \in \mathbb{R}^{I_1 \times I_1}$ and $\mathbf{R}_1^{(n)} \in \mathbb{R}^{I_1 \times I_1}$ have to be stored, totaling $\mathcal{O}(I_1^2)$ entries (their triangular or orthonormal structures can be exploited to save storage space). Updating of the evolving mode 3 requires the storage of the matrices $\mathbf{Q}_3^{(n)} \in \mathbb{R}^{w \times I_1 I_2}$, $\mathbf{R}_3^{(n)} \in \mathbb{R}^{I_1 I_2 \times I_1 I_2}$ and $\mathbf{V}_3^{(n)} \in \mathbb{R}^{I_1 I_2 \times I_1 I_2}$, with $\mathcal{O}(wI_1 I_2 + I_1^2 I_2^2)$ entries in total. For the three modes together, $\mathcal{O}(wI_1 I_2 + I_1^2 I_2^2)$ entries have to be stored, not taking the slices in non-local memory into account, which are stored until they are eventually downdated. Since no compression has been considered, this is not smaller than the $\mathcal{O}(I_1 I_2 w)$ entries of the full tensor. In Section IV, we will consider strategies to reduce the storage cost.

TABLE I
MEMORY AND COMPUTATIONAL COMPLEXITY FOR THE BATCH TMLSVD METHOD AND THE PROPOSED UPDATING METHOD. SUPERSCRIPTS ARE DROPPED AND $R = R_1^{(n)} = R_2^{(n)} = R_3^{(n)}$ IS ASSUMED.

Memory		
Batch		$\mathcal{O}(I_1 I_2 w)$
Updating		
	New slice	$\mathcal{O}(I_1 I_2)$
	Slices in window	$\mathcal{O}(wI_1 I_2)$
	Modes 1 and 2	$\mathcal{O}(I_1^2 + I_2^2)$
	Mode 3	$\mathcal{O}(I_1^2 I_2^2 + wI_1 I_2)$
Computation		
Batch		$\mathcal{O}(I_1 I_2 w \min(I_1, I_2) + I_1 I_2 w R)$
Updating		
	Modes 1 and 2	$\mathcal{O}(I_1^2 I_2 + I_2^2 I_1)$
	Mode 3	$\mathcal{O}(I_1 I_2 (w + I_1 I_2))$
	Core	$\mathcal{O}(R^2 I_1 I_2)$

2) Computational complexity: For mode 1, the complexity of one subspace update is similar to that of a single QR-update ($\mathcal{O}(I_1^2)$ flop). Of course, every new slice adds I_2 columns to $\mathbf{T}_{[1;3,2]}^{(n)}$, while removing one slice corresponds to removing I_2 columns from $\mathbf{T}_{[1;3,2]}^{(n)}$, so the total complexity of one slice update or downdate for mode 1 is $\mathcal{O}(I_1^2 I_2)$.

For the evolving mode 3, the update consists of computing the vector $\mathbf{x}^{\mathrm{T}} = \mathbf{t}^{\mathrm{T}}\mathbf{V}$ and triangularizing the $\mathbf{R}$-matrix, which require $\mathcal{O}(I_1^2 I_2^2)$ and $\mathcal{O}(wI_1 I_2 + I_1^2 I_2^2)$ flop, respectively. The complexity of the QR downdate is $\mathcal{O}(wI_1 I_2)$ flop (the matrix $\mathbf{V}_{\text{up}}$ is not involved in the downdate). Finally, making the URV decomposition rank-revealing requires one deflation step, at a cost of $\mathcal{O}(I_1^2 I_2^2)$ flop [10], [11]. If the core tensor is required, it can be computed in $\mathcal{O}(I_1 I_2 R^2)$ flop, assuming $R = R_1^{(n)} = R_2^{(n)} = R_3^{(n)}$. The total complexity is thus $\mathcal{O}(I_1 I_2 (w + I_1 I_2))$ flop. The results from this section are summarized in Table I.

D. Choosing the threshold values γ_i

The SURV decomposition adjusts the dimension of the subspace in the ith mode automatically through the value of the threshold γ_i. Following [12] and [24], we use a threshold value of $\gamma_i = \alpha\sigma(\sqrt{m} + \sqrt{n})$, wherein σ is an estimate of the standard deviation of the i.i.d. additive Gaussian noise and m and n are the dimensions of $\mathbf{T}_{(i)}$ after applying the sliding window. For example, for mode 1, $m = I_1$ and $n = wI_2$ and for the evolving mode 3, one has $m = w$ and $n = I_1 I_2$. Intuitively, $\sigma(\sqrt{m} + \sqrt{n})$ forms an estimate of the largest singular value of an $(m \times n)$-matrix consisting of only noise [24]. Consequently, the expected value of the largest noise component in $\mathbf{T}_{(i)}$ is equal to $\sigma(\sqrt{m} + \sqrt{n})$. In practice, we set the noise threshold higher to avoid incorrect rank estimations due to randomness, i.e., the largest noise components could be slightly higher than $\sigma(\sqrt{m} + \sqrt{n})$. The

parameter α can thus be seen as a safety factor. If α is chosen too close to 1, noise components could accidentally make it into the signal subspace if they are larger than expected due to chance. Increasing α limits this risk. On the other hand, if α is chosen too large, small subspace components could be considered as noise, which is also undesirable. In practice, the range of suitable values is often quite large, especially when the signal-to-noise ratio is sufficiently high. A value of 1.24 was proposed by the authors of [24] as a trade-off between risking to incorrectly include noise components and risking to incorrectly discard signal components. This value also performed well in our experiments, so we used it throughout.

III. Error bounds for the TMLUTVD

The truncation of the MLUTVD by discarding the information in the non-principal subspaces generally introduces an error. However, we can bound this truncation error by looking at the norms of the parts of the triangular matrices that are discarded [12]. Below, we adapt the Frobenius and spectral norms to an updating context.

A. Frobenius norm

Following [12], we can bound the squared TMLUTVD error in the Frobenius norm ϵ_{F}^2 in function of the truncated parts in every mode, as shown in Figure 6. The exact error for all unfoldings can be computed from the $\mathbf{L}_i$ and $\mathbf{R}$-matrices of the SURV and RRURV decompositions, respectively. The following theorem gives an expression for $\|\mathbf{T}^{(n)}_{[1;3,2]} - \mathbf{U}^{(n)}_1 \mathbf{U}^{(n)\mathrm{T}}_1 \mathbf{T}^{(n)}_{[1;3,2]}\|^2_{\mathrm{F}}$, the error that is introduced by discarding the orthogonal complement of the principal subspace in the SURV decomposition for mode 1.

Theorem 1 (Approximation bound for non-evolving modes): Let $\mathbf{Q}_1\mathbf{L}_1\mathbf{J}\mathbf{\Phi}_1\mathrm{T}$ be an SURV decomposition of $[\mathbf{N}_1\ \mathbf{T}_{[1;3,2]}]\mathbf{J}$. Split $\mathbf{Q}_1 \in \mathbb{R}^{I_1 \times I_1}$ into $\mathbf{Q}_1 = [\mathbf{Q}_N\ \mathbf{Q}_S]$, wherein $\mathbf{Q}_S \in \mathbb{R}^{I_1 \times R_1}$ is a basis for the principal column space and $\mathbf{Q}_N \in \mathbb{R}^{I_1 \times (I_1 - R_1)}$ is a basis for the orthogonal complement. Partition $\mathbf{\Phi}_1$ into $\mathbf{\Phi}_1 = \left[\begin{smallmatrix} \mathbf{\Phi}_{11} & \mathbf{\Phi}_{12} \\ \mathbf{\Phi}_{21} & \mathbf{\Phi}_{22} \end{smallmatrix}\right]$ such that $\mathbf{\Phi}_{11}$ and $\mathbf{\Phi}_{22}$ are square and their dimensions match the column dimensions of $\mathbf{N}_1$ and $\mathbf{T}_{[1;3,2]}$, respectively. Finally, partition $\mathbf{L}_1$ into $\mathbf{L}_1 = \left[\begin{smallmatrix} \mathbf{L}_{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{L}_{21} & \mathbf{0}\ \mathbf{L}_{22} & \mathbf{0} \end{smallmatrix}\right]$, with $\mathbf{L}_{11}$ and $\mathbf{L}_{22}$ square, such that it matches the structure of $\mathbf{Q}_1$ and $\mathbf{\Phi}_1$. Then

$$\left\|\mathbf{T}_{[1;3,2]} - \mathbf{Q}_S\mathbf{Q}_S^{\mathrm{T}}\mathbf{T}_{[1;3,2]})\right\|^2_{\mathrm{F}} = \gamma_1^2(I_1 - R_1) - \|\mathbf{L}_{11}\|^2_{\mathrm{F}}.$$

Proof: See Appendix A. ∎

Here, $\|\mathbf{L}_{11}\|^2_{\mathrm{F}}$ is the squared Frobenius norm of the part that is truncated from the SURV decomposition, equaling the sum of the squares of the singular values σ_l, for $l = 1, 2 \ldots, (I_1 - R_1)$, of $\mathbf{L}_{11}$. However, the SURV considers a shifted decomposition, i.e., it looks at the eigenvalues of $\mathbf{T}_{[1;3,2]}\mathbf{T}^{\mathrm{T}}_{[1;3,2]} - \mathbf{N}_1\mathbf{N}_1^{\mathrm{T}}$, instead of those of $\mathbf{T}_{[1;3,2]}\mathbf{T}^{\mathrm{T}}_{[1;3,2]}$. The true squared singular values of the truncated part are thus $\gamma_1^2 - \sigma_l^2$, for $l = 1, 2 \ldots, (I_1 - R_1)$, leading to the result from Theorem 1.

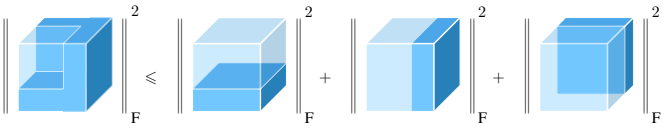


Fig. 6. The squared Frobenius norm of the TMLUTVD error is upper-bounded by the sum of the truncation errors in the different modes. As discussed in Section III-B, a similar inequality holds for the spectral norm.

Mode 2 can be analogously bounded and for mode 3, we track an RRURVD and directly use the bound from [12]. Combining the mode-n bounds gives an upper bound for ϵ_{F}^2:

$$\epsilon_{\mathrm{F}}^2 \leq \gamma_1^2(I_1 - R_1) - \left\|\mathbf{L}^{(n)}_{1_{11}}\right\|^2_{\mathrm{F}} + \gamma_2^2(I_2 - R_2) - \left\|\mathbf{L}^{(n)}_{2_{11}}\right\|^2_{\mathrm{F}} + \left\|\mathbf{G}^{(n)}_3\right\|^2_{\mathrm{F}},$$

wherein $\mathbf{L}_1$ and $\mathbf{L}_2$ refer to the mode-1 and 2 $\mathbf{L}$-matrices.

B. Spectral norm

For ϵ_2, the error in the spectral norm [26], defined as

$$\|\mathcal{A}\|_2 = \max_{\|\mathbf{x}\|=\|\mathbf{y}\|=\|\mathbf{z}\|=1} \mathcal{A} \cdot_1 \mathbf{x} \cdot_2 \mathbf{y} \cdot_3 \mathbf{z},$$

we can write:

$$\epsilon_2 \leq \gamma_1 + \gamma_2 + \gamma_3,$$

following [12]. For the spectral norm, the shift of the SURV decomposition does not affect the bound, as it only shifts each σ_m^2 to $\gamma_i^2 - \sigma_m^2$. The result is that for SURV the maximal error γ_i is achieved when a particular singular value σ_l of $\mathbf{L}_{i11}$ is equal to zero, while for the RRURVD, it occurs when $\sigma_r = \gamma_i$, with σ_r a singular value of $\mathbf{G}_3$.

IV. Strategies to reduce the algorithm complexity

In the previous sections, we outlined the basic TMLUTVD updating algorithm. While it allows us to obtain a new TMLUTVD from the previous decomposition with provably bounded approximation error, it is not very efficient with respect to memory and computational cost. Below, we provide three strategies to improve the efficiency of the algorithm. The first strategy can be used when the window length is small compared to the product of the dimensions of the non-evolving modes 1 and 2 and comes at no cost to the accuracy of the method. The second strategy is to first compute a low-rank approximation of the new slice. Using this approximation instead of the full slice can greatly reduce the number of SURV updates that have to be performed; the bounds from Section III can be adapted to include the approximation error of the new slice (details are omitted). The final strategy is to truncate sequentially, i.e., immediately after processing each mode, like for the sequentially truncated MLSVD [6] (STMLSVD). With this strategy, modes that are processed later, such as the evolving mode 3, require less work than when truncation is performed after all mode-n subspaces have been computed. If the mode-1 and mode-2 subspaces are stationary, the bounds of Section III can be modified similarly to how the MLSVD bounds [5] are adapted to the STMLSVD [6]. If they are non-stationary, however, the situation is more involved,

as previously truncated information cannot be recovered if it is needed later (within the next $(w-1)$ updates). Thus, potentially, the bounds form Section III need to be increased to account for this information loss. If the ambient space is slowly-varying, though, the information loss is minor and the accuracy of the method is generally not significantly impacted, see Section V. In Table II, the storage and computational costs are shown using the strategies from this section.

A. Exploiting small window lengths

If $w < I_1 I_2$, tracking an ambient space of dimensions $I_1 I_2$ for the evolving mode 3 is unnecessary, as the dimension of the mode-3 subspace can never exceed w. Instead, one can consider a space of dimension w and extract the principal mode-3 subspace of dimension R_3 from it without loss of accuracy. More concretely, one can track an RRURVD $\mathbf{Q}_3^{(n)}\mathbf{R}_3^{(n)}\mathbf{V}_3^{(n)^{\mathrm{T}}}$ of the mode-3 unfolding of $\mathcal{T}^{(n)}$, with $\mathbf{Q}_3^{(n)} \in \mathbb{R}^{w\times w}$, $\mathbf{R}_3^{(n)} \in \mathbb{R}^{w\times w}$ and $\mathbf{V}_3^{(n)} \in \mathbb{R}^{I_1 I_2 \times w}$. As the window length w is smaller than $I_1 I_2$ in this case, the computational cost of updating and downdating this decomposition is reduced by the use of the smaller RRURV matrices. The only drawback is that the rank-revealing structure of the $\mathbf{R}$-matrix can be lost and up to $R_3^{(n)}$ deflation steps may be required to restore this structure [10]. This approach leads to a computational complexity for the evolving mode of $\mathcal{O}(w(wR_3 + I_1 I_2))$ flop: the update and downdate require $\mathcal{O}(wI_1 I_2)$ and $\mathcal{O}(w^2)$ flop, respectively, when using the smaller RRURV matrices and restoring the rank-revealing structure with at most R_3 deflation steps on a $(w \times w)$-matrix costs $\mathcal{O}(w^2 R_3)$ flop. A more detailed exposition is given below.

a) Update: After adding a new row $\mathbf{t}^{\mathrm{T}}$ to the unfolded tensor $\mathbf{T}_{[3;2,1]}^{(n)}$, one can explicitly construct the following decomposition of the augmented mode-3 unfolding of $\mathcal{T}$:

$$\mathbf{T}_{[3;2,1]}^{(n+1)} = \begin{bmatrix} \mathbf{T}_{[3;2,1]}^{(n)} \\ \mathbf{t}^{\mathrm{T}} \end{bmatrix} \approx \begin{bmatrix} \mathbf{Q}_3^{(n)} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_3^{(n)} & \mathbf{0} \\ \mathbf{x}^{\mathrm{T}} & y \end{bmatrix} \begin{bmatrix} \mathbf{V}_3^{(n)^{\mathrm{T}}} \\ \mathbf{v}^{\mathrm{T}} \end{bmatrix}. \quad (3)$$

The vector $\mathbf{x}^{\mathrm{T}}$ from (3) is obtained as the coefficients of the projection of the new row $\mathbf{t}^{\mathrm{T}}$ on the space spanned by the rows of $\mathbf{V}_3^{(n)^{\mathrm{T}}}$. The vector $y\mathbf{v}^{\mathrm{T}}$, with $\|\mathbf{v}\| = 1$ holds the part of $\mathbf{t}^{\mathrm{T}}$ that lies in the orthogonal complement. Writing $\mathbf{V}_{\text{up}} = [\mathbf{V}_3^{(n)} \quad \mathbf{v}] \in \mathbb{R}^{I_1 I_2 \times (w+1)}$, one obtains:

$$\mathbf{T}_{[3;2,1]}^{(n+1)} \approx \begin{bmatrix} \mathbf{Q}_3^{(n)} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_3^{(n)} & \mathbf{0} \\ \mathbf{x}^{\mathrm{T}} & y \end{bmatrix} \mathbf{V}_{\text{up}}^{\mathrm{T}}.$$

The new middle matrix can overall be brought back to upper-triangular form by zeroing the entries in its last row $[\mathbf{x}^{\mathrm{T}}\ y]$ from left to right with Givens row rotations [25]. The inverse rotations are applied as column rotations to the left-most matrix to obtain the new URV decomposition $\mathbf{Q}_{\text{up}}\mathbf{R}_{\text{up}}\mathbf{V}_{\text{up}}^{\mathrm{T}}$.

b) Downdate: The downdate of the oldest row of $\mathbf{T}_{[3;2,1]}^{(n)}$ does not depend on the $\mathbf{V}$-matrix, so we can proceed as in Section II-B with a QR downdate [11]. The rank-revealing structure of the $\mathbf{R}$-matrix is then restored with deflation steps. The new $\mathbf{R}$- and $\mathbf{V}$-matrices now have $(w+1)$ columns instead of w, so after eliminating one column, we obtain the new URV decomposition $\mathbf{Q}_3^{(n+1)}\mathbf{R}_3^{(n+1)}\mathbf{V}_3^{(n+1)^{\mathrm{T}}}$. As before, the first $R_3^{(n+1)}$ columns of $\mathbf{Q}_3^{(n+1)}$ form a basis for the new mode-3 subspace and are thus used to construct $\mathbf{U}_3^{(n+1)} \in \mathbb{R}^{w \times R_3^{(n+1)}}$.

B. Compressing the new slice

To update the mode-1 subspace of the tensor with an entire new slice $\mathbf{T}_{n+1}$, I_2 SURV updates are required. If I_2 is large, this makes a slice update computationally costly, even though the SURV algorithm itself is quite efficient, see Section II-C2. However, if the mode-2 rank $R_2^{(n)}$ of the LMLRA from the previous update is small compared to I_2, it is reasonable to expect the new slice to have low column rank as well. It can then be computationally efficient to first compute a low-rank approximation of rank $p^{(n+1)}$ of the new slice $\mathbf{T}_{n+1}$. Instead of applying an SURV update for all I_2 columns of $\mathbf{T}_{n+1}$, one can thus update the mode-1 subspace with the $p^{(n+1)}$ columns of the approximation. The lower number of SURV updates more than compensates for the one-time $\mathcal{O}(I_1 I_2 p^{(n+1)})$ flop cost of computing the approximation and the complexity for the mode-1 update drops from $\mathcal{O}(I_1^2 I_2)$ to $\mathcal{O}(I_1^2 p^{(n+1)})$ flop. When $p^{(n+1)} = \mathcal{O}(R_1^{(n+1)})$ is small compared to I_2, this offers a considerable efficiency gain. The threshold for the low-rank approximation can be set as $\epsilon = \delta\gamma_1$, with $\delta < 1$, in accordance with the SURV threshold value γ_1. By choosing $\delta = 0.1$, for instance, subspace vectors that do not lie in the principal mode-1 subspace of the new slice, but that may be significant enough after accumulation over multiple slice updates, are not immediately discarded. The same low-rank approximation can be used for the mode-2 SURV update.

The low-rank approximation of the slice can also be used to make the downdate less costly. Old slices are stored until they are dropped from the window, thus at any point, w $(I_1 \times I_2)$-matrices have to be stored. Fortunately, we can reuse the low-rank approximation of the old slice that was computed w time steps before during its update. This can significantly lower the memory requirements of the method, as only a rank-$p^{(n+1)}$ approximation of each slice is stored for w time steps. This approximation $\hat{\mathbf{T}}_{::n-w}$ is then used to downdate the mode-1 and mode-2 subspaces. Using only the $p^{(n+1)}$ columns of the low-rank approximation directly lowers the computational cost of the slice downdates. There is no overhead of the computation of the approximation, as it is already available.

C. Sequential truncation

Instead of first computing all principal subspaces and then truncating, we can also truncate immediately after updating each mode, as for the STMLSVD [6]. This has the advantage that the new slice becomes smaller as more modes are processed and the computational cost for the later modes is decreased. Below, we explain how sequential truncation can be applied to the TMLUTVD updating process. Using this approach and the compression from Section IV-B, the total complexity of the TMLUTVD update can be reduced to $\mathcal{O}(R^2(\min(R^4, w^2) + I^2 R))$ flop, where $R = R_1 = R_2 = R_3$ and $i = I_1 = I_2$ are assumed, see Table II.

1) Non-evolving modes 1 *and* 2*:* After updating the mode-1 subspace, the newly computed principal subspace $\text{col}(\mathbf{U}_1^{(n+1)})$ can be applied to reduce the computational cost of the subspace updates in the following modes. With a matrix projection of the new slice onto the mode-1 principal subspace, the number of columns of $\mathbf{T}_{n+1}^{\mathrm{T}}$ can be reduced from I_1 to $R_1^{(n+1)}$. We compute $\mathbf{T}_{n+1}^{\mathrm{T}}\mathbf{U}_1^{(n+1)} \in \mathbb{R}^{I_2 \times R_1^{(n+1)}}$ and the $R_1^{(n+1)}$ columns of the resulting matrix are then used to perform $R_1^{(n+1)}$, instead of I_1, SURV updates for mode 2. After obtaining the new mode-2 factor matrix $\mathbf{U}_2^{(n+1)}$ from the SURV updates, this matrix can be used to further project $\mathbf{T}_{n+1}^{\mathrm{T}}\mathbf{U}_1^{(n+1)}$, now to the mode-2 principal subspace:

$$\tilde{\mathbf{T}}_{n+1} = {\mathbf{U}_1^{(n+1)}}^{\mathrm{T}}\mathbf{T}_{n+1}\mathbf{U}_2^{(n+1)} = \mathbf{T}_{n+1} \cdot_1 {\mathbf{U}_1^{(n+1)}}^{\mathrm{T}} \cdot_2 {\mathbf{U}_2^{(n+1)}}^{\mathrm{T}}.$$

The matrix $\tilde{\mathbf{T}}_{n+1} \in \mathbb{R}^{R_1^{(n+1)} \times R_2^{(n+1)}}$ can then be used to make the update of the evolving mode 3 more efficient, see Section IV-C2 below. For the downdate of the oldest slice in the window, the same approach can be used as for the update to decrease the computational cost.

For sequential truncation, the updates of the non-evolving modes can be ordered strategically to minimize the computational cost. We compare the operation counts of both orderings. We find that updating mode 2 after mode 1 is faster if $I_1^2 I_2 + I_2^2 R_1^{(n+1)} < I_2^2 I_1 + I_1^2 R_2^{(n+1)}$. Writing S_i for the dimension reduction in mode i, we have $S_1 = I_1 - R_1^{(n+1)}$ and $S_2 = I_2 - R_2^{(n+1)}$, and the comparison becomes $I_1^2 I_2 + I_2^2(I_1 - S_1) < I_2^2 I_1 + I_1^2(I_2 - S_2)$, or $I_2^2 S_1 > I_1^2 S_2$. When $S_1 \approx S_2$, the mode with the smallest dimension should thus be updated first. This ordering is twice beneficial, as it generally leads to a relatively low approximation error as well [6].

2) Evolving mode 3*:* In Section II-B for mode 3, we track an RRURVD $\mathbf{Q}_3^{(n)}\mathbf{R}_3^{(n)}{\mathbf{V}_3^{(n)}}^{\mathrm{T}}$ of the mode-3 unfolding of $\mathcal{T}^{(n)}$. The matrix $\mathbf{V}_3^{(n)}$ is an $(I_1 I_2 \times I_1 I_2)$-matrix, so storage of this matrix, and computations with it, are problematic for large tensors. Fortunately, we can work with a $\mathbf{V}$-matrix that has $R_1^{(n+1)}R_2^{(n+1)}$, instead of $I_1 I_2$, columns by using the sequentially truncated new slice $\tilde{\mathbf{T}}_{n+1} \in \mathbb{R}^{R_1^{(n+1)} \times R_2^{(n+1)}}$ instead of the full new slice. This significantly decreases the storage and computational cost of the mode-3 update.

Due to the sequential truncation of $\mathcal{T}^{(n)}$ in modes 1 and 2, we can consider the row space of $\mathbf{T}_{[3;2,1]}^{(n)}$ to be spanned by the columns of $\mathbf{U}_2^{(n)} \otimes \mathbf{U}_1^{(n)}$ with dimension $R_1^{(n)}R_2^{(n)}$. Thus, an RRURVD of $\mathbf{T}_{[3;2,1]}^{(n)}$ can be written as follows:

$$\mathbf{T}_{[3;2,1]}^{(n)} \approx \mathbf{Q}_3^{(n)}\mathbf{R}_3^{(n)}{\tilde{\mathbf{V}}_3^{(n)}}^{\mathrm{T}}(\mathbf{U}_2^{(n)} \otimes \mathbf{U}_1^{(n)})^{\mathrm{T}},$$

wherein the matrix $(\mathbf{U}_2^{(n)} \otimes \mathbf{U}_1^{(n)})\tilde{\mathbf{V}}_3^{(n)}$ takes the role of the matrix $\mathbf{V}_3^{(n)}$. Herein $\tilde{\mathbf{V}}_3^{(n)} \in \mathbb{R}^{R_1^{(n)}R_2^{(n)} \times R_1^{(n)}R_2^{(n)}}$ requires much less memory to store than $\mathbf{V}_3^{(n)} \in \mathbb{R}^{I_1 I_2 \times I_1 I_2}$. Another advantage of this approach is that the updated core tensor is obtained without much computational effort: its the mode-3 unfolding equals $\mathbf{R}_{3_{1:R_3,1:R_3}}^{(n)} {\tilde{\mathbf{V}}_{3_{:,1:R_3}}^{(n)}}^{\mathrm{T}} \in \mathbb{R}^{R_3 \times R_1 R_2}$.

Below, we describe the update and downdate for the evolving mode in detail. We assume $w \geq R_1^{(n)}R_2^{(n)}$, but sequential truncation can be combined with the strategies from Section IV-A if $w < R_1^{(n)}R_2^{(n)}$.

a) Update: After computing the new mode-1 and mode-2 subspaces, the RRURVD for mode-3 is adapted to take these subspace changes into account before including the contributions of the new slice $\tilde{\mathbf{T}}_{n+1}$. We construct a decomposition

$$\mathbf{T}_{[3;2,1]}^{(n+1)} \approx \mathbf{Q}_3^{(n)}\mathbf{R}_3^{(n)}\mathbf{P}^{\mathrm{T}}(\mathbf{U}_2^{(n+1)} \otimes \mathbf{U}_1^{(n+1)})^{\mathrm{T}},$$

introducing the matrix $\mathbf{P} \in \mathbb{R}^{R_1^{(n+1)}R_2^{(n+1)} \times R_1^{(n)}R_2^{(n)}}$, which is an updated version of the matrix $\tilde{\mathbf{V}}_3^{(n)}$ that takes the new mode-1 and mode-2 subspaces into account:

$$\mathbf{P} = (\mathbf{U}_2^{(n+1)} \otimes \mathbf{U}_1^{(n+1)})^{\mathrm{T}}(\mathbf{U}_2^{(n)} \otimes \mathbf{U}_1^{(n)})\tilde{\mathbf{V}}_3^{(n)}. \tag{4}$$

It is possible to avoid the explicit computation of the Kronecker product in (4) by applying the elementary rotations that transform $\mathbf{U}_1^{(n)}$ and $\mathbf{U}_2^{(n)}$ into $\mathbf{U}_1^{(n+1)}$ and $\mathbf{U}_2^{(n+1)}$, respectively, directly on the tensorization of $\tilde{\mathbf{V}}_3^{(n)}$. Afterwards, we orthogonalize $\mathbf{P}$ through the computation of a QL-decomposition $\mathbf{P} = \mathbf{Q}'\mathbf{L}'$ to obtain:

$$\mathbf{T}_{[3;2,1]}^{(n+1)} \approx \mathbf{Q}_3^{(n)}\mathbf{R}_3^{(n)}{\mathbf{L}'}^{\mathrm{T}}{\mathbf{Q}'}^{\mathrm{T}}(\mathbf{U}_2^{(n+1)} \otimes \mathbf{U}_1^{(n+1)})^{\mathrm{T}}.$$

Taking $\mathbf{R}_3^{(n)}{\mathbf{L}'}^{\mathrm{T}} \in \mathbb{R}^{R_1^{(n)}R_2^{(n)} \times R_1^{(n+1)}R_2^{(n+1)}}$ as the new $\mathbf{R}$-matrix[1] and $\mathbf{Q}' \in \mathbb{R}^{R_1^{(n+1)}R_2^{(n+1)} \times R_1^{(n+1)}R_2^{(n+1)}}$ as the new $\tilde{\mathbf{V}}$-matrix, we now proceed as in Section II-B to include the contribution of the new slice $\tilde{\mathbf{T}}_{n+1}$ into the decomposition $\mathbf{Q}_{\text{up}}\mathbf{R}_{\text{up}}\tilde{\mathbf{V}}_{\text{up}}^{\mathrm{T}}(\mathbf{U}_2^{(n+1)} \otimes \mathbf{U}_1^{(n+1)})^{\mathrm{T}}$. As $\tilde{\mathbf{T}}_{n+1}$ is already compressed to dimensions $R_1^{(n+1)} \times R_2^{(n+1)}$, we can ignore the matrix $(\mathbf{U}_2^{(n+1)} \otimes \mathbf{U}_1^{(n+1)})^{\mathrm{T}}$ for this step and work only with the small matrix $\tilde{\mathbf{V}}_{\text{up}} \in \mathbb{R}^{R_1^{(n+1)} \times R_2^{(n+1)} \times R_1^{(n+1)} \times R_2^{(n+1)}}$ which is very convenient computationally.

b) Downdate: The downdate in Section II-B is independent of $\mathbf{V}_{\text{up}}$, so downdating $\mathbf{Q}_{\text{up}}\mathbf{R}_{\text{up}}\tilde{\mathbf{V}}_{\text{up}}^{\mathrm{T}}(\mathbf{U}_2^{(n+1)} \otimes \mathbf{U}_1^{(n+1)})^{\mathrm{T}}$ into $\mathbf{Q}_{\text{dn}}\mathbf{R}_{\text{dn}}\tilde{\mathbf{V}}_{\text{dn}}^{\mathrm{T}}(\mathbf{U}_2^{(n+1)} \otimes \mathbf{U}_1^{(n+1)})^{\mathrm{T}}$ can be done in the same way. If the mode-1 and mode-2 subspaces are mildly non-stationary, the rank-revealing property of $\mathbf{R}_{\text{dn}}$ can be restored by a small number of deflation steps [10] to obtain $\mathbf{Q}_3^{(n+1)}\mathbf{R}_3^{(n+1)}\tilde{\mathbf{V}}_3^{(n+1)^{\mathrm{T}}}(\mathbf{U}_2^{(n+1)} \otimes \mathbf{U}_1^{(n+1)})^{\mathrm{T}}$. In highly non-stationary environments, up to $R_3^{(n+1)}$ steps may be required.

V. Experiments

In a first experiment, the LMLRA of a tensor is tracked throughout a series of abrupt subspace changes. Next, we perform tracking in a DOA estimation context. We also compare the speed and accuracy of the TMLUTVD tracking method to a TMLSVD tracking algorithm. Finally, we demonstrate how TMLUTVD tracking can speed up CPD tracking in a chemical application. The experiments are performed on a computer with an Intel Core i7-6820HQ CPU at 2.70GHz and 16GB of RAM using MATLAB R2016b and Tensorlab 3.0 [27].

[1]Note that some columns right of the diagonal are nonzero in the QL-decomposition of the wide matrix $\mathbf{P}$ in case that $R_1^{(n+1)}R_2^{(n+1)} < R_1^{(n)}R_2^{(n)}$. Additionally, the new $\mathbf{R}$-matrix has more rows than columns, so one or more of its least significant rows, and the corresponding columns of $\mathbf{Q}_3^{(n)}$, should be eliminated. Consequently, slightly more work is required in this case before the update can be performed.

TABLE II
WITH THE STRATEGIES FROM SECTION IV, TMLUTVD UPDATING REQUIRES LESS STORAGE AND IS FASTER THAN THE BATCH ALGORITHM. STORAGE AND COMPUTATIONAL COMPLEXITY FOR THE BATCH TMLSVD METHOD AND THE PROPOSED UPDATING METHOD. SUPERSCRIPTS ARE DROPPED AND $R = R_1^{(n)} = R_2^{(n)} = R_3^{(n)}$ IS ASSUMED.

Memory		
Batch		$\mathcal{O}(I_1 I_2 w)$
Updating		
	New slice	$\mathcal{O}(I_1 I_2)$
	Slices in window	$\mathcal{O}(w(I_1 + I_2)p)$
	Modes 1 and 2	$\mathcal{O}(I_1^2 + I_2^2)$
	Mode 3	$\mathcal{O}(\min(w, R^2)(w + \min(w, R^2) + R^2))$
Computation		
Batch		$\mathcal{O}(I_1 I_2 w \min(I_1, I_2) + I_1 I_2 w R)$
Updating		
	Modes 1 and 2	$\mathcal{O}((I_1 + I_2) I_1 p + I_2^2 R)$
	Mode 3	$\mathcal{O}(R^2(\min(w^2, R^4) + I_1 + I_2))$
	Core	$\mathcal{O}(R^4)$

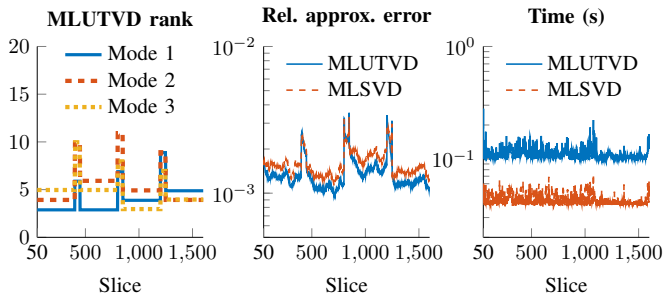


Fig. 7. The updating method tracks an LMLRA of a tensor with high accuracy, even with abrupt changes in the subspaces. Updating of the LMLRA of four low rank $(100 \times 100 \times 400)$-tensors, concatenated in the third mode, with a window length of 50. Left: Estimated rank values by the TMLUTVD tracking method for all three modes. Center: Relative Frobenius error for TMLUTVD tracking and batch TMLSVD. Right: Computation time per update for TMLUTVD tracking and batch TMLSVD.

A. Updating with abrupt subspace changes

Four low multilinear rank tensors of dimensions $100 \times 100 \times 400$ are concatenated in the third mode, simulating a situation with abrupt subspace changes. The subtensors have multilinear rank $(3, 4, 5)$, $(3, 6, 5)$, $(4, 5, 3)$ and $(5, 4, 4)$, respectively. The factor matrix entries and the core are drawn from the uniform distribution on $[0, 1]$ and Gaussian distributed noise with an SNR of 30dB is added to the tensor. The decomposition is initialized and updated with a window length of 50. In Figure 7, the estimated dimensions of the principal subspaces are shown, as are the relative Frobenius errors with respect to the windowed tensor and the computation times for both the basic TMLUTVD tracking algorithm (without the strategies from Section IV) and the batch TMLSVD method. The main reason for the comparison with the batch method is that the batch version is, at this point, actually *the* reference method, i.e., it allows for the estimation of the correct multilinear rank in every step and shows the best accuracy one can hope for. Both methods use the thresholds from Section II-D and $\alpha = 1.24$ [24]. The latter computes the MLSVD of the windowed tensor from scratch after every update, without sequential truncation (to be consistent with the TMLUTVD method). As seen in Figure 7, the correct dimensions of the principal subspaces are found throughout the whole updating process and the approximation error is lower for the TMLUTVD tracking method than for the batch TMLSVD method. The lower error is a result of the better SURV subspace estimate of the mode-1 and mode-2 subspaces, which is consistent with observations in [24]. When the sliding window moves into a new low multilinear rank tensor and thus covers two low multilinear rank tensors partially, the estimated mode-n rank values are equal to the sums of the mode-n rank values of the two tensors in the window, which is as expected. When the last slice of the old low multilinear rank tensor is downdated, the estimated mode-n rank values become equal to those of the most recent low multilinear rank tensor. By decreasing the window length, the TMLUTVD method can be made to react quicker to abrupt subspace changes, but smaller window lengths typically lead to larger approximation errors, as shown

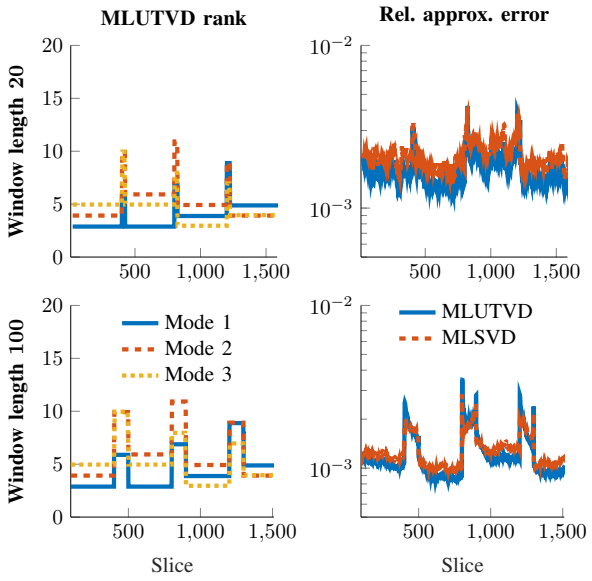


Fig. 8. Shorter window lengths lead to faster reaction times to abrupt subspace changes, but at the cost of a larger approximation error. Updating of the LMLRA of four low rank $(100 \times 100 \times 400)$-tensors, concatenated in the third mode, with a window length of 20 (top) and 100 (bottom). Left: Estimated rank values by the TMLUTVD tracking method for all three modes. Right: Relative Frobenius error for TMLUTVD tracking and batch TMLSVD.

in Figure 8. Looking at Figure 7 (right), the TMLUTVD tracking method is slower than the batch TMLSVD method, as the computational complexity depends on $I_1^2 I_2^2$ instead of $I_1 I_2 w$. However, it has the advantage that the full tensor is not processed in every time instance: only the newest and oldest slices are used.

In Figure 9, the experiment is repeated using both compression and sequential truncation, see Section IV-B and IV-C. The computation time of the updating method is significantly decreased and the STMLUTVD tracking method is now about as fast as the batch STMLSVD method, while the accuracy is unaffected. Note that SVD computation is highly optimized in MATLAB, in contrast to SURV and RRURVD computation, so the time comparison is very biased against our algorithm.

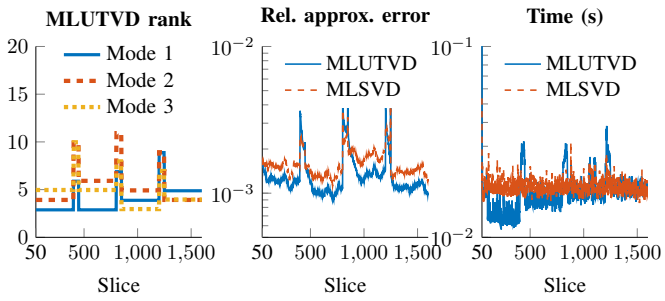


Fig. 9. By compressing the new slices (Section IV-B) and using sequential truncation (Section IV-C), the method can be sped up significantly, without negatively impacting the accuracy. Same experiment setting as in Figure 7.

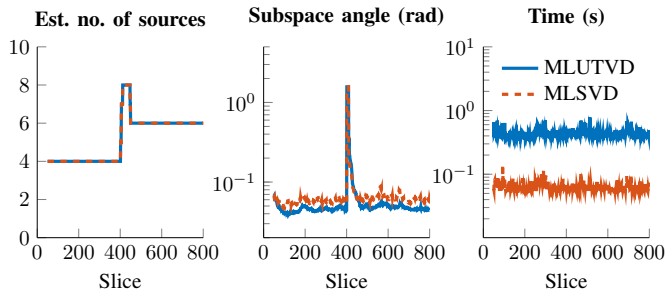


Fig. 10. The TMLUTVD updating method tracks the DOAs of several sources with high accuracy and correctly estimates the number of sources. For the first and last 400 time instances, four and six sources are tracked, respectively. Left: Estimated mode-3 rank values. Center: Angle in radians between the column spaces of the true and estimated unfolded steering tensors. Right: Computation time per update for TMLUTVD updating and batch TMLSVD.

B. Direction-of-arrival estimation

We estimate the DOAs of a number of sources impinging from the far-field on a uniform rectangular array (URA), using a data model similar to the one in [28]. We consider a set of R undamped exponentials impinging on a URA of dimensions $K \times L$ for N time instances. Because of the URA structure, the observation tensor $\mathcal{Y} \in \mathbb{C}^{K \times L \times N}$ has entries

$$y_{kln} = \sum_{r=1}^{R} s_r(n) \mathrm{e}^{j(k-1)\mu_{r1}} \mathrm{e}^{j(l-1)\mu_{r2}} + n_{kln},$$

with j the imaginary unit, n_{kln} noise, $s_r(n)$ the complex amplitude of the rth exponential at the nth time instance and μ_r the coordinates of the rth spatial frequency vector. The latter coordinates depend on the DOAs of the R sources. This means that we can model $\mathcal{Y}$ as follows:

$$\mathcal{Y} = \mathcal{A} \cdot_3 \mathbf{S}^{\mathrm{T}} + \mathcal{N},$$

wherein $\mathbf{S} \in \mathbb{C}^{R \times N}$ holds the source signals, $\mathcal{N}$ holds noise and $\mathcal{A} \in \mathbb{C}^{K \times L \times R}$ is the array steering tensor. We have

$$\mathbf{A}_{::r} = \mathbf{a}^{(1)}(\mu_{r1})(\mathbf{a}^{(2)}(\mu_{r2}))^{\mathrm{T}},$$

wherein $\mathbf{a}^{(1)}(\mu_{r1}) = \left[1 \ \mathrm{e}^{j\mu_{r1}} \ \mathrm{e}^{2j\mu_{r1}} \ \ldots \ \mathrm{e}^{(K-1)j\mu_{r1}}\right]^{\mathrm{T}} \in \mathbb{C}^{K}$, and $\mathbf{a}^{(2)}(\mu_{r2}) \in \mathbb{C}^{L}$ is defined analogously. From this model, we can derive that, in the absence of noise, $\mathcal{Y}$ has rank R and that the different mode-n rank values of $\mathcal{Y}$ do not exceed R.

From the data model above, a TMLSVD/TMLUTVD can be used to recover the signal subspace with a higher accuracy than with matrix-based methods [28]. One computes $\mathcal{Y} \approx \mathcal{C} \cdot_1 \mathbf{U}_1 \cdot_2 \mathbf{U}_2 \cdot_3 \mathbf{U}_3$ and compares the column space of the estimated unfolded steering tensor $\hat{\mathbf{A}}_{[2,1;3]}$, with $\hat{\mathcal{A}} = \mathcal{C} \cdot_1 \mathbf{U}_1 \cdot_2 \mathbf{U}_2 \in \mathbb{C}^{K \times L \times R_3^{(n)}}$, with the column space of the true unfolded steering tensor $\mathbf{A}_{[2,1;3]}$. The comparison is performed through the computation of the subspace angle between these subspaces. Using the proposed TMLUTVD tracking method, we can track the steering tensor and account for changes in the number of sources or their location.

We consider a URA of dimensions $K \times L$, with $K = L = 80$. For the first 400 time instances, we generate $R = 4$ sources with azimuth and elevation (in degrees) equal to $\left[45 \ -45 \ 35 \ 75\right]$ and $\left[45 \ -45 \ 15 \ 85\right]$, respectively. For the next 400 time instances, sources three and four no longer appear and four new sources are added in their place. We thus have $R = 6$ sources with azimuth and elevation equal to $\left[45 \ -45 \ 65 \ 15 \ 35 \ -65\right]$ and $\left[45 \ -45 \ -5 \ 25 \ 85 \ -50\right]$, respectively. Both the real and imaginary parts of the source signals are sampled from the standard normal distribution and we add additive Gaussian noise with an SNR of $0\,\mathrm{dB}$ to both parts of the observed samples. We then track the signal subspaces using the TMLUTVD tracking method with a window length of 50 time instances and compare the approach to batch TMLSVD, again with the thresholds as in Section II-D and $\alpha = 1.24$. The results can be seen in Figures 10 and 11 for the basic TMLSVD/TMLUTVD approach and for the approach with compression and sequential truncation (see Section IV), respectively. Aside from the error on the estimation of the column space of the steering matrix (center plot), we also show the estimated mode-3 rank, corresponding to the estimated number of sources, and the time per update for both methods. From both figures, it can be seen that the TMLUTVD tracking method finds the correct number of sources and achieves at least the same accuracy as batch TMLSVD. As for the first experiment, the former method is actually more accurate than the latter. When the sources change abruptly, both methods needs some time to correctly estimate the new steering matrix. This is as expected for a sliding window approach: the original sources only disappear from the window after 50 new updates have been processed. From Figures 10 and 11, we can clearly see that compression and sequential truncation can speed up the tracking method dramatically with minimal effect on the accuracy of the subspace estimation.

C. Comparison with TMLSVD updating

Batch TMLSVD methods achieve the maximum accuracy in the estimation of the principal mode-n subspaces of a tensor, but they have a high computational cost. Faster approaches exist in the form of TMLSVD updating methods, at the cost of a loss in accuracy. In TMLSVD updating methods, SVDs are tracked for each mode through a sliding window [17], building on classical SVD updating approaches [29]–[31]. These methods require the maximal tracked multilinear rank to be set at the start, so all information that is not included in the tracked TMLSVD is lost (in contrast to our proposed algorithm.) In Figure 12, we compare our proposed TMLUTVD updating

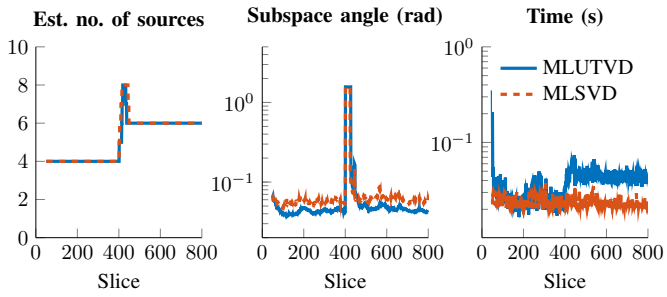


Fig. 11. With compression (Section IV-B) and sequential truncation (Section IV-C), the TMLUTVD tracking method is about as fast as the batch STMLSVD algorithm, with minimal loss in accuracy for the estimation of the subspaces. Same experiment setting as in Figure 10.

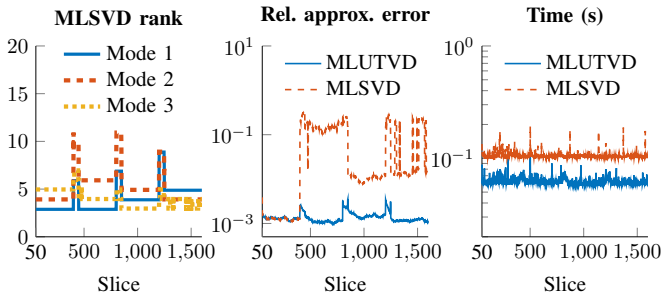


Fig. 12. The TMLUTVD updating method is more accurate and faster than the TMLSVD updating method in the experiment from Subsection V-A. Updating of the LMLRA of four low rank $(100 \times 100 \times 400)$-tensors, concatenated in the third mode, with a window length of 50. Left: Estimated rank values by the TMLSVD tracking method for all three modes. Center: Relative Frobenius error for TMLUTVD and TMLSVD tracking. Right: Computation time per update for TMLUTVD and TMLSVD tracking.

method with an TMLSVD updating method. We applied a hybrid approach for TMLSVD updating, using the method from [30] to track the SVD of the evolving mode and the method from [31] for the non-evolving modes. We used a maximal mode-n rank of 15 for each of the tensor modes. In Figure 12, it can be seen that our proposed method outperforms the TMLSVD updating algorithm, both in accuracy and speed. In the left plot, one can see that the TMLSVD updating method occasionally picks an incorrect mode-n rank, e.g., around slice 1500, in contrast to our TMLUTVD updating method (shown in Figure 7).

D. Hazardous gasses

Orthogonal compression is one of the most important applications of LMLRAs. It is a standard preprocessing step in the computation of other tensor decompositions such as the CPD or the block term decomposition (BTD) [32] and may significantly decrease the computation time [33]. Even in the computation of constrained CPDs or BTDs, orthogonal compression can be used to one's computational advantage [34]. In a nutshell, the idea is as follows. An LMLRA $\mathcal{T} = \mathcal{C} \cdot_1 \mathbf{U}^{(1)} \cdot_2 \mathbf{U}^{(2)} \cdot_3 \mathbf{U}^{(3)}$ *compresses* $\mathcal{T}$ into a core tensor $\mathcal{C}$ with dimensions equal to those of the principal mode-n subspaces and next, a CPD of the core tensor is computed, with factor matrices $\mathbf{V}^{(1)}$, $\mathbf{V}^{(2)}$ and $\mathbf{V}^{(3)}$, denoted by $[\![\mathbf{V}^{(1)}, \mathbf{V}^{(2)}, \mathbf{V}^{(3)}]\!]$. The CPD of $\mathcal{T}$ is then given by $[\![\mathbf{U}^{(1)}\mathbf{V}^{(1)}, \mathbf{U}^{(2)}\mathbf{V}^{(2)}, \mathbf{U}^{(3)}\mathbf{V}^{(3)}]\!]$. Here we extend this approach to updating and tracking scenarios, exploiting the features of the TMLUTVD. Namely, we consider applications where a CPD of a large tensor is to be tracked or slice-by-slice computation of the CPD is required due to time or memory constraints. Generally, CPD updating methods, e.g., [35], [36] are used in these cases to track the CPD or build the CPD incrementally. However, it can be useful to track a TMLUTVD and to estimate the CPD of its time-evolving core in every step or at the end. In this approach, we use the tracked TMLUTVD as a compact proxy of the actual data, with a model accuracy that can be set by the user (as explained in Section III). If the TMLUTVD approximation error is bounded by ϵ_{F}^2 and the core tensor $\mathcal{C}$ admits a CPD up to an error of $\tilde{\epsilon}_{\mathrm{F}}^2 := \|\mathcal{C} - [\![\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3]\!]\|_{\mathrm{F}}^2$, then the CPD $[\![\mathbf{U}_1\mathbf{V}_1, \mathbf{U}_2\mathbf{V}_2, \mathbf{U}_3\mathbf{V}_3]\!]$ is guaranteed to fit the given data tensor $\mathcal{T}$ with an error bounded by $\epsilon_{\mathrm{F}}^2 + \tilde{\epsilon}_{\mathrm{F}}^2$. Updating allows us to find the CPD of very large tensors with a certificate of accuracy. This has the additional benefit that the multilinear rank found by the tracked TMLUTVD can help to find a suitable rank for the CPD, i.e., the rank is lower-bounded by the multilinear rank.

As an illustration, we adapt the chemo-sensing application from [37] to the MLUTVD updating setting. The goal of the experiment is predicting which hazardous gas is detected by a sensor array. The original dataset consists of ten analytes, measured at six positions in a wind tunnel under different wind conditions and temperatures: 18000 time series of 260s sampled at 100Hz were collected for each of the 72 sensors [38]. The same preprocessing steps are performed as in [37] (selecting three of the ten gasses, trimming start and end of the time series, interpolating missing values, mean subtraction, etc.) and the data is additionally downsampled by a factor of 25, resulting in a tensor with dimensions $1036 \times 72 \times 899$, with the three modes representing time, sensor and experiment, respectively. To determine the specific hazardous gas that is measured, a CPD of rank 5 is computed. The resulting components in the experiment mode can distinguish between the different hazardous gasses, see Figure 13. Details about the actual classification can be found in [37].

In our adapted chemical experiment, we run through the (downsampled) tensor slice by slice to build the CPD incrementally and we compare two approaches. The first is the CPD updating method from [35]. The second is tracking an LMLRA by performing MLUTVD updating (with compression and sequential truncation) followed by a CPD of the resulting core tensor in the final time step. As initialization for the CPDs, we use factor matrices drawn from a standard normal distribution. As stopping criteria for the CPD algorithm (`cpd_nls` from Tensorlab [27]), we use a relative function value improvement of less than 10^{-8} and no more than 100 iterations. During the CPD updating process, we initialize the factor matrices with those found in the previous update (and use the least-squares estimate for the new row at the bottom for the evolving mode, see [35]). Here, we use a maximum of 10 iterations. For the MLUTVD threshold values, we use a standard deviation of 10^{-3} (determined experimentally) and $\alpha = 1.24$, as before. The accuracy and computation time for

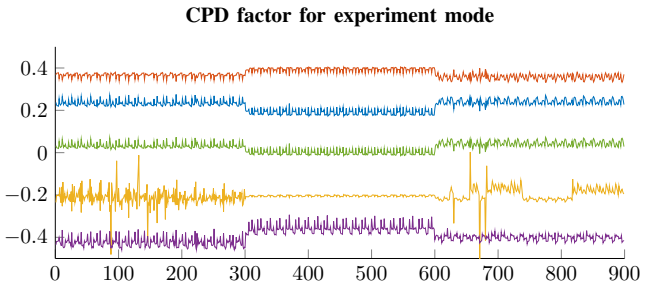


Fig. 13. The five columns of the CPD factor matrix in the experiment mode for the chemical sensing tensor. Note that the experiments have been ordered according to their specific hazardous gas and they have been shifted vertically away from 0 with increments of 0.2 for visual purposes. This clearly shows three different patterns on the left/middle/right corresponding to the three hazardous gasses.

TABLE III
WITH MLUTVD UPDATING, THE RESULTING CPD IS SIGNIFICANTLY MORE ACCURATE THAN WHEN CPD UPDATING IS USED, AND THE COMPUTATION TIME IS LOWER. COMPARISON BETWEEN MLUTVD UPDATING, FOLLOWED BY A CPD OF THE FINAL CORE TENSOR, AND CPD UPDATING FOR A CHEMICAL DATA TENSOR THAT IS PROCESSED SLICE BY SLICE. BOTH THE TOTAL COMPUTATION TIME AND THE RELATIVE FROBENIUS ERROR ARE SHOWN FOR BOTH APPROACHES.

	Time	Rel. approx. error
MLUTVD updating	264 s	0.26434
CPD updating	671 s	0.29747

both methods can be seen in Table III. One can see that the MLUTVD updating approach is not only more accurate than the CPD updating method, it is also significantly faster. By tracking a TMLUTVD, we can store as much information from the tensor as desired (by adjusting the thresholds) and thus control the accuracy and the speed of the resulting CPD computation.

VI. CONCLUSION

An updating method was proposed to track an LMLRA of a tensor as new slices are added and removed in a certain mode. The algorithm uses a TMLUTVD to track the subspaces of both the evolving and non-evolving modes with automatic rank detection in a numerically stable way. We showed that the approximation error of the tracked LMLRA can be bounded and derived the complexities with respect to storage and computation. Several strategies were proposed to significantly speed up the algorithm. We showed that the algorithm can reduce the cost to compute an LMLRA compared to the batch TMLSVD algorithm, with limited loss in accuracy. An experiment showed that our proposed algorithm can cope with abrupt subspace changes. Next, we showed that TMLUTVD updating can outperform TMLSVD updating. Finally, we extended the standard preprocessing step that implements an orthogonal compression in batch CPD algorithms to the setting of CPD updating and tracking.

APPENDIX A
APPROXIMATION BOUND FOR NON-EVOLVING MODES

Theorem 1 (Approximation bound for non-evolving modes): Let $\mathbf{Q}_1\mathbf{L}_1\mathbf{J}\mathbf{\Phi}_1^{\mathrm{T}}$ be an SURV decomposition of $[\mathbf{N}_1 \;\; \mathbf{T}_{[1;3,2]}]\mathbf{J}$. Split $\mathbf{Q}_1 \in \mathbb{R}^{I_1 \times I_1}$ into $\mathbf{Q}_1 = [\mathbf{Q}_N \;\; \mathbf{Q}_S]$, wherein $\mathbf{Q}_S \in \mathbb{R}^{I_1 \times R_1}$ is a basis for the principal column space and $\mathbf{Q}_N \in \mathbb{R}^{I_1 \times (I_1 - R_1)}$ is a basis for the orthogonal complement. Partition $\mathbf{\Phi}_1$ into $\mathbf{\Phi}_1 = \left[\begin{smallmatrix} \mathbf{\Phi}_{11} & \mathbf{\Phi}_{12} \\ \mathbf{\Phi}_{21} & \mathbf{\Phi}_{22} \end{smallmatrix}\right]$ such that $\mathbf{\Phi}_{11}$ and $\mathbf{\Phi}_{22}$ are square and their dimensions match the column dimensions of $\mathbf{N}_1$ and $\mathbf{T}_{[1;3,2]}$, respectively. Finally, partition $\mathbf{L}_1$ into $\mathbf{L}_1 = \left[\begin{smallmatrix} \mathbf{L}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{L}_{21} & \mathbf{0} & \mathbf{L}_{22} & \mathbf{0} \end{smallmatrix}\right]$, with $\mathbf{L}_{11}$ and $\mathbf{L}_{22}$ square, such that it matches the structure of $\mathbf{Q}_1$ and $\mathbf{\Phi}_1$. Then

$$\left\|\mathbf{T}_{[1;3,2]} - \mathbf{Q}_S\mathbf{Q}_S^{\mathrm{T}}\mathbf{T}_{[1;3,2]})\right\|_{\mathrm{F}}^2 = \gamma_1^2(I_1 - R_1) - \|\mathbf{L}_{11}\|_{\mathrm{F}}^2 .$$

Proof: For simplicity, we will give the proof for an infinite length sliding window. First, we write $\left[\begin{smallmatrix} \overline{\mathbf{L}}_{11} & \mathbf{0} \\ \overline{\mathbf{L}}_{21} & \overline{\mathbf{L}}_{22} \end{smallmatrix}\right] := \left[\begin{smallmatrix} \mathbf{L}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{L}_{21} & \mathbf{0} & \mathbf{L}_{22} & \mathbf{0} \end{smallmatrix}\right]$ by extending $\mathbf{L}_{11}$, $\mathbf{L}_{21}$ and $\mathbf{L}_{22}$ with zero columns. Then, we replace $\mathbf{T}_{[1;3,2]}$ in the expression for the Frobenius norm of the mode-1 approximation error by its decomposition $-\mathbf{Q}_1\mathbf{L}_1\left[\mathbf{\Phi}_{21} \;\; -\mathbf{\Phi}_{22}\right]^{\mathrm{T}} = \mathbf{Q}_1\mathbf{L}_1\left[-\mathbf{\Phi}_{21} \;\; \mathbf{\Phi}_{22}\right]^{\mathrm{T}}$, wherein the minuses come from the multiplications with $\mathbf{J}$:

$$\begin{aligned}
&\left\|\mathbf{T}_{[1;3,2]} - \mathbf{Q}_S\mathbf{Q}_S^{\mathrm{T}}\mathbf{T}_{[1;3,2]}\right\|_{\mathrm{F}}^2 = \left\|\mathbf{Q}_N\mathbf{Q}_N^{\mathrm{T}}\mathbf{T}_{[1;3,2]}\right\|_{\mathrm{F}}^2 \\
&= \left\|\mathbf{Q}_N\mathbf{Q}_N^{\mathrm{T}}\mathbf{Q}_1\mathbf{L}_1\left[-\mathbf{\Phi}_{21} \;\; \mathbf{\Phi}_{22}\right]^{\mathrm{T}}\right\|_{\mathrm{F}}^2 \\
&= \left\|\mathbf{Q}_N\mathbf{Q}_N^{\mathrm{T}}((\mathbf{Q}_N\overline{\mathbf{L}}_{11} + \mathbf{Q}_S\overline{\mathbf{L}}_{21})(-\mathbf{\Phi}_{21}^{\mathrm{T}}) + \mathbf{Q}_S\overline{\mathbf{L}}_{22}\mathbf{\Phi}_{22}^{\mathrm{T}})\right\|_{\mathrm{F}}^2 \\
&= \left\|-\mathbf{Q}_N\overline{\mathbf{L}}_{11}\mathbf{\Phi}_{21}^{\mathrm{T}}\right\|_{\mathrm{F}}^2 = \left\|\overline{\mathbf{L}}_{11}\mathbf{\Phi}_{21}^{\mathrm{T}}\right\|_{\mathrm{F}}^2 ,
\end{aligned}$$

wherein we used that $\mathbf{Q}_N^{\mathrm{T}}\mathbf{Q}_S = \mathbf{0}$ and $\mathbf{Q}_N^{\mathrm{T}}\mathbf{Q}_N = \mathbf{I}$ due to the orthogonality of $\mathbf{Q}_1$. As $\|\mathbf{A}\|_{\mathrm{F}}^2 = \mathrm{tr}(\mathbf{A}^{\mathrm{T}}\mathbf{A}) = \mathrm{tr}(\mathbf{A}\mathbf{A}^{\mathrm{T}})$ for any matrix $\mathbf{A}$ and $\mathbf{\Phi}_{21}^{\mathrm{T}}\mathbf{\Phi}_{21} = \mathbf{\Phi}_{11}^{\mathrm{T}}\mathbf{\Phi}_{11} - \mathbf{I}$ due to the J-orthogonality of $\mathbf{\Phi}$, we can write

$$\begin{aligned}
&\left\|\overline{\mathbf{L}}_{11}\mathbf{\Phi}_{21}^{\mathrm{T}}\right\|_{\mathrm{F}}^2 = \mathrm{tr}(\overline{\mathbf{L}}_{11}\mathbf{\Phi}_{21}^{\mathrm{T}}\mathbf{\Phi}_{21}\overline{\mathbf{L}}_{11}^{\mathrm{T}}) \\
&= \mathrm{tr}(\overline{\mathbf{L}}_{11}(\mathbf{\Phi}_{11}^{\mathrm{T}}\mathbf{\Phi}_{11} - \mathbf{I})\overline{\mathbf{L}}_{11}^{\mathrm{T}}) \\
&= \mathrm{tr}(\overline{\mathbf{L}}_{11}\mathbf{\Phi}_{11}^{\mathrm{T}}\mathbf{\Phi}_{11}\overline{\mathbf{L}}_{11}^{\mathrm{T}}) - \mathrm{tr}(\overline{\mathbf{L}}_{11}\overline{\mathbf{L}}_{11}^{\mathrm{T}}) \\
&= \left\|\overline{\mathbf{L}}_{11}\mathbf{\Phi}_{11}^{\mathrm{T}}\right\|_{\mathrm{F}}^2 - \left\|\overline{\mathbf{L}}_{11}\right\|_{\mathrm{F}}^2 = \left\|\overline{\mathbf{L}}_{11}\mathbf{\Phi}_{11}^{\mathrm{T}}\right\|_{\mathrm{F}}^2 - \|\mathbf{L}_{11}\|_{\mathrm{F}}^2 . \quad (5)
\end{aligned}$$

Analogously, we find, using that $\mathbf{N}_1\mathbf{N}_1^{\mathrm{T}} = \gamma_1^2\mathbf{I}$:

$$\begin{aligned}
&\left\|\mathbf{N}_1 - \mathbf{Q}_S\mathbf{Q}_S^{\mathrm{T}}\mathbf{N}_1\right\|_{\mathrm{F}}^2 = \left\|\mathbf{Q}_N\mathbf{Q}_N^{\mathrm{T}}\mathbf{N}_1\right\|_{\mathrm{F}}^2 = \left\|\overline{\mathbf{L}}_{11}\mathbf{\Phi}_{11}^{\mathrm{T}}\right\|_{\mathrm{F}}^2 \\
&= \mathrm{tr}(\mathbf{Q}_N^{\mathrm{T}}\mathbf{N}_1\mathbf{N}_1^{\mathrm{T}}\mathbf{Q}_N) = \gamma_1^2\left\|\mathbf{Q}_N\right\|_{\mathrm{F}}^2 = \gamma_1^2(I_1 - R_1). \quad (6)
\end{aligned}$$

Combining (5) and (6) gives us

$$\left\|\mathbf{T}_{[1;3,2]} - \mathbf{Q}_S\mathbf{Q}_S^{\mathrm{T}}\mathbf{T}_{[1;3,2]})\right\|_{\mathrm{F}}^2 = \gamma_1^2(I_1 - R_1) - \|\mathbf{L}_{11}\|_{\mathrm{F}}^2$$

■

REFERENCES

[1] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, 2017.

[2] A. Cichocki, C. Mandic, A.-H. Phan, C. Caiafa, G. Zhou, Q. Zhao, and L. De Lathauwer, "Tensor decompositions for signal processing applications. From two-way to multiway component analysis," *IEEE Signal Process. Mag.*, vol. 32, pp. 145–163, 2015.

[3] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *J. Math. Phys.*, vol. 6, no. 1–4, pp. 164–189, 1927.

[4] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychom.*, vol. 31, no. 3, pp. 279–311, 1966.

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

[6] N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen, "A new truncation strategy for the higher-order singular value decomposition," *SIAM J. Sci. Comput.*, vol. 34, no. 2, pp. A1027–A1052, 2012.

[7] J. P. Delmas, *Subspace Tracking for Signal Processing*, ch. 4, pp. 211–270. John Wiley & Sons, Ltd, 2010.

[8] L. De Lathauwer and J. Vandewalle, "Dimensionality reduction in higher-order signal processing and rank-$(R_1, R_2, \ldots, R_N)$ reduction in multilinear algebra," *Linear Algebr. its Appl.*, vol. 391, pp. 31–55, 11 2004.

[9] M. Vandecappelle and L. De Lathauwer, "Low multilinear rank updating," in *Proceedings of the 53rd Asilomar Conference on Signals, Systems and Computers (ASILOMAR 2019)*, (Pacific Grove, CA, USA), pp. 437–441, Nov 2019.

[10] R. D. Fierro and P. C. Hansen, "Low-rank revealing UTV decompositions," *Numer. Algorithms*, vol. 15, no. 1, pp. 37–55, 1997.

[11] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD: Johns Hopkins University Press, 4th ed., 2013.

[12] M. Vandecappelle and L. De Lathauwer, "From multilinear SVD to multilinear UTV decomposition," *Signal Process.*, vol. 198, p. 108575, 2022.

[13] B. Yang, "An extension of the PASTd algorithm to both rank and subspace tracking," *IEEE Signal Process. Lett.*, vol. 2, pp. 179–182, Sep. 1995.

[14] R. Badeau, B. David, and G. Richard, "Fast approximated power iteration subspace tracking," *IEEE Trans. Signal Process.*, vol. 53, pp. 2931–2941, Aug 2005.

[15] W. Hu, X. Li, X. Zhang, X. Shi, S. Maybank, and Z. Zhang, "Incremental tensor subspace learning and its applications to foreground segmentation and tracking," *Int. J. Comput. Vis.*, vol. 91, pp. 303–327, Feb 2011.

[16] X. Ma, D. Schonfeld, and A. Khokhar, "Dynamic updating and downdating matrix SVD and tensor HOSVD for adaptive indexing and retrieval of motion trajectories," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009)*, (Taipei, Taiwan), pp. 1129–1132, April 2009.

[17] A. Sobral, C. G. Baker, T. Bouwmans, and E.-h. Zahzah, "Incremental and multi-feature tensor subspace learning applied for background modeling and subtraction," in *International Conference on Image Analysis and Recognition*, pp. 94–103, Springer, 2014.

[18] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos, "Incremental tensor analysis: Theory and applications," *ACM Trans. Knowl. Discov. from Data*, vol. 2, pp. 11:1–11:37, Oct 2008.

[19] Y. Sun, Y. Guo, C. Luo, J. Tropp, and M. Udell, "Low-rank Tucker approximation of a tensor from streaming data," *SIAM J. Math. Data Sci.*, vol. 2, no. 4, pp. 1123–1150, 2020.

[20] O. A. Malik and S. Becker, "Low-rank Tucker decomposition of large tensors using tensorsketch," *Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[21] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Rev.*, vol. 53, no. 2, pp. 217–288, 2011.

[22] Y. Cheng, F. Roemer, O. Khatib, and M. Haardt, "Tensor subspace tracking via Kronecker structured projections (TeTraKron) for time-varying multidimensional harmonic retrieval," *EURASIP J. Adv. Signal Process.*, vol. 2014:123, pp. 1–14, 11 2014.

[23] N. Vannieuwenhoven, K. Meerbergen, and R. Vandebril, "Computing the gradient in optimization algorithms for the CP decomposition in constant memory through tensor blocking," *SIAM J. Sci. Comput.*, vol. 37, no. 3, pp. C415–C438, 2015.

[24] M. Zhou and A. van der Veen, "Stable subspace tracking algorithm based on a Signed URV decomposition," *IEEE Trans. Signal Process.*, vol. 60, pp. 3036–3051, June 2012.

[25] G. W. Stewart, "An updating algorithm for subspace tracking," *IEEE Trans. Signal Process.*, vol. 40, pp. 1535–1541, 1992.

[26] S. Friedland and L.-H. Lim, "Nuclear norm of higher-order tensors," *Math. Comput.*, vol. 87, no. 311, pp. 1255–1281, 2018.

[27] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer, "Tensorlab 3.0," 3 2016.

[28] M. Haardt, F. Roemer, and G. Del Galdo, "Higher-order SVD-based subspace estimation to improve the parameter estimation accuracy in multidimensional harmonic retrieval problems," *Signal Process. IEEE Trans.* , vol. 56, pp. 3198 – 3213, 2008.

[29] C. G. Baker, K. A. Gallivan, and P. Van Dooren, "Low-rank incremental methods for computing dominant singular subspaces," *Linear Algebr. its Appl.*, vol. 436, no. 8, pp. 2866–2888, 2012.

[30] A. Levy and M. Lindenbaum, "Sequential Karhunen-Loeve basis extraction and its application to images," in *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No. 98CB36269)*, vol. 2, pp. 456–460, IEEE, 1998.

[31] R. Badeau, G. Richard, and B. David, "Sliding window adaptive SVD algorithms," *IEEE Trans. Signal Process.*, vol. 52, no. 1, pp. 1–10, 2004.

[32] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms — Part II: Definitions and uniqueness," *SIAM J. Matrix Analysis Appl. (SIMAX)*, vol. 30, pp. 1033–1066, 9 2008.

[33] J. D. Carrol, S. Pruzansky, and J. B. Kruskal, "CANDELINC: A general approach to multidimensional analysis of many-way arrays with linear constraints on parameters," *Psychom.*, vol. 45, pp. 3–24, 3 1980.

[34] N. Vervliet, O. Debals, and L. De Lathauwer, "Exploiting efficient representations in large-scale tensor decompositions," *SIAM J. Sci. Comput.*, vol. 41, pp. A789–A815, 3 2019.

[35] M. Vandecappelle, N. Vervliet, and L. De Lathauwer, "Nonlinear least squares updating of the canonical polyadic decomposition," in *Proceedings of the 25th European Signal Processing Conference (EUSIPCO 2017, Kos, Greece)*, pp. 693–697, 9 2017.

[36] D. Nion and N. Sidiropoulos, "Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor," *IEEE Trans. Signal Process.*, vol. 57, pp. 2299–2310, 6 2009.

[37] N. Vervliet and L. De Lathauwer, "A randomized block sampling approach to canonical polyadic decomposition of large-scale tensors," *IEEE J. Sel. Top. Signal Process.*, vol. 10, pp. 284–295, 2016.

[38] A. Vergara, J. Fonollosa, J. Mahiques, M. Trincavelli, N. Rulkov, and R. Huerta, "On the performance of gas sensor arrays in open sampling systems using inhibitory support vector machines," *Sensors Actuators, B: Chem.*, vol. 185, pp. 462–477, 2013.