A Novel Recursive Least-Squares Adaptive Method For Streaming Tensor-Train Decomposition With Incomplete Observations

Thanh Trung Le[a,b], Karim Abed-Meriam[a,c], Nguyen Linh Trung[b,*], Adel Hafiane[a]

[a] *University of Orleans, INSA-CVL, PRISME, France*
[b] *Vietnam National University, Hanoi, VNU-UET, Vietnam*
[c] *Academic Institute of France, France*

Abstract

Tensor tracking which is referred to as online (adaptive) decomposition of streaming tensors has recently gained much attention in the signal processing community due to the fact that many modern applications generate a huge number of multidimensional data streams over time. In this paper, we propose an effective tensor tracking method via the tensor-train format for decomposing high-order incomplete streaming tensors. On the arrival of new data, the proposed algorithm minimizes a weighted least-squares objective function accounting for both missing values and time-variation constraints on the underlying tensor-train cores, thanks to the recursive least-squares filtering technique and the block coordinate descent framework. Our algorithm is fully capable of tensor tracking from noisy, incomplete, and high-dimensional observations in both static and time-varying environments. Its tracking ability is validated with several experiments on both synthetic and real data.

Keywords: Tensor-train, tensor decomposition, adaptive algorithms, online algorithms, streaming data, missing data

1. Introduction

Tensor-train (TT) decomposition, which is one form of tensor decomposition, has become a powerful processing tool for multi-dimensional and large-scale data analysis [1]. Under the TT format, we can factorize a high-order tensor into a sequence of 3rd-order tensors. TT

*Corresponding author
Email address: linhtrung@vnu.edu.vn (Nguyen Linh Trung)

decomposition offers several advantages compared to the two standard CP/PARAFAC decomposition, Tucker decomposition, and their combination called block-term decomposition (BTD). For example, we can represent any high-order tensor under TT decomposition. It is due to the fact that the existence of the best low-rank tensor approximation with fixed rank parameter is always guaranteed [2, 3]. It is also indicated in [4] that the TT-rank coincides with the separation rank of the underlying tensor.[1] In other words, the TT-rank is a uniquely defined quantity and it can be effectively determined in a stable way. Moreover, TT decomposition provides a memory-saving representation for high-order tensors and can break the curse of dimensionality which limits the order of the tensors to be analysed [2, 5]. Accordingly, TT decomposition is expected to be capable of handling big tensors efficiently and effectively. We refer the readers to [1] for a comprehensive survey on basic properties, algorithms, and applications of the TT decomposition.

In recent years, the demand for big data stream analysis has been increasing rapidly [6]. In most modern online applications, data acquisition is a time-varying process where data are sequentially acquired at a large scale with many attributes over time. This leads to several issues for tensor decomposition in general and TT decomposition in particular: (i) size of the tensor is growing linearly with time, (ii) time variation in nonstationary environments where the underlying process generating the tensor can change over time, and (iii) uncertainties (e.g., imprecise, noisy, and misleading entries) emanate during data collection, to name a few. In parallel, missing data are ubiquitous in multi-dimensional and large-scale data analysis where collecting all data attributes at a time is either too expensive or even impossible due to corruption [7]. Accordingly, it is of great interest to develop adaptive/online/streaming tensor decomposition or tensor tracking algorithms which are capable of handling these issues. In spite of several successes in batch settings, TT decomposition has not gained the same popularity in online settings as CP and Tucker decompositions, see [8] for a good review on tensor tracking algorithms. Particularly, most of the existing TT methods are operating in batch-mode and become inefficient for streaming applications.

Related Works: There exist few TT methods related to adaptive tensor decomposition in the literature. In [9, 10, 11], Lubich *et al.* introduced some dynamical tensor approximation methods under TT format for factorizing time-varying tensors, thanks to the Dirac–Frenkel-McLachlan variational principle. However, the dynamical tensors of in-

[1] Denote by s_i the rank of the n-th unfolding matrix of $\boldsymbol{\mathcal{X}}$ of order N. The vector $\mathbf{s} = [s_1, s_2, \ldots, s_N]$ is called the separation rank of $\boldsymbol{\mathcal{X}}$.

terest are of fixed size, and hence, their methods indeed belong to the class of batch TT algorithms. In [12], Liu *et al.* proposed an incremental TT method called iTTD for decomposing high-order tensors of which one dimension grows with time. iTTD factorizes new streams as individual tensors into TT-cores and then appends the estimated cores to old estimates from past observations. In [13], Wang *et al.* also developed an incremental TT method for factorizing tensors derived from industrial IoT data streams, namely AITT. By exploiting a relationship between the directly reshaped matrix and integration of unfolding matrices, AITT can estimate effectively the underlying TT-cores with low cost. Nevertheless, it is worth noting that the framework of both iTTD and AITT is not really online streaming learning, but incremental batch learning. Recently, we have introduced an adaptive TT method called TT-FOA capable of tracking the low-rank components of high-order tensors in online settings [14]. Its design is useful for scenarios where a single data sample is acquired at a time. Despite advantages in some respects, all the existing dynamical/incremental/adaptive TT methods above are not suitable for handling streaming tensors with missing values. In parallel, many tensor completion algorithms have been proposed in the literature [15]. They are, however, either batch tensor methods and not useful for stream processing or deployed under other tensor formats (e.g., CP, Tucker, BTD, and t-SVD).

Apart from the mentioned favorable characteristics such as rank determination, stable computation, and memory-saving representation, TT and its variants (e.g., block TT [16], cyclic TT or tensor chain/ring [17], fully-connected tensor network [18], and tensor wheel [19]) offer more flexible low-rank representations for streaming tensors compared to classical tensor formats. TT can be easily transformed into CP, Tucker, and BTD formats [1, Sec. 4.3]. For instance, TT simplifies to CP when the 3rd-order TT-cores have diagonal lateral slices. By applying appropriate regularization and constraints on core tensors, adaptive TT methods can efficiently factorize streaming tensors, benefiting from the advantageous properties of CP, Tucker, and BTD. Another appealing feature of the streaming TT model is that it admits several mathematical and graphical representations that can be interchanged for different applications. Through tensor network transformations like contractions, reshaping, and decompositions of core tensors, TT can be converted into a tensor ring and vice versa [1, Sec. 2.2]. In the tensor ring representation, core tensors are interconnected circularly and treated equivalently [17]. Consequently, adaptive TT methods can effectively handle streaming tensors with any time-varying mode or dimension, unlike classical adaptive tensor de-

compositions designed for a specific streaming mode. Additionally, TT exhibits adaptability to diverse signal processing models, such as multidimensional harmonic retrieval, canonical correlation analysis and tracking extreme singular values/vectors in SVD [20]. This flexibility of TT in modeling high-dimensional data is therefore noteworthy. Furthermore, TT and its adaptive variant provide natural sparse and distributed representations for large tensors, effectively addressing both established and emerging methodologies for tensor-based representations and optimization [1, 21]. With the increasing need to handle large-scale and high-speed streaming data, including tensors, this feature becomes highly beneficial [8].

Main Contribution: In this paper, we propose a novel adaptive algorithm called ATT (which stands for Adaptive Tensor-Train) for decomposing high-order incomplete streaming tensors with time under the tensor-train format. By utilizing the recursive least-squares method in adaptive filtering, ATT minimizes effectively a weighted least-squares objective function accounting for both missing values and time-variation constraints on the underlying tensor-train cores. The proposed ATT algorithm is scalable, effective, and capable of estimating low-rank components of streaming tensors from noisy and incomplete observations as well as tracking their time variation in nonstationary environments. To the best of our knowledge, ATT is the first of its kind which is capable of dealing with time-dependent streaming tensors with missing values.

Compared to the state-of-the-art TT methods, ATT presents a novel optimization approach with several appealing features. Unlike classical batch TT decomposition methods (e.g., TT-SVD [2] and TT-HSVD [22]), which rely on computationally expensive SVD decompositions of unfolding matrices, ATT takes a different route. Specifically, ATT employs a recursive least-squares filtering technique that involves performing only simple matrix-vector multiplications and inverse operations on small-sized matrices (their size is equal to the TT rank), thereby resulting in significantly reduced computational complexity. One key advantage of ATT is its ability to update each TT-core independently, without interfering with the others. This characteristic facilitates parallel and distributed computing, making it highly advantageous for dealing with large-scale and higher-order tensors. In comparison to existing adaptive tensor decomposition methods, ATT leverages second-order estimation instead of the commonly adopted first-order estimation in several online tensor decomposition methods such as TT-FOA [14], TeCPSGD [23], and OLCP [24]. Thanks to its efficient recursive procedure, which eliminates the need for inverting the main Hessian matrix, ATT achieves a computational complexity similar to that of first-order estimation methods. This

approach, coupled with a novel regularization term on TT-cores, enables ATT to effectively and efficiently handle missing observations and time-varying data in online settings. Moreover, ATT's update rule can be designed to operate at the row-wise level, offering extensive support for parallel and distributed processing. This design choice leads to accelerated tracking, particularly in cases of significant data corruption. For instance, ATT allows for skipping the update of specific rows in the TT-core during updates, without affecting others. This not only enhances computational efficiency but also contributes to ATT's remarkable effectiveness in handling missing data.

Notations: Lowercase, boldface lowercase, boldface capital, and bold calligraphic letters are used to denote scalars (e.g., a), vectors (e.g., $\mathbf{a}$), matrices (e.g., $\mathbf{A}$), and tensors (e.g., $\boldsymbol{\mathcal{A}}$), respectively. We use $\mathbf{A}^{-1}$, $\mathbf{A}^{\top}$, and $\mathbf{A}^{-\top}$ to represent the inverse of $\mathbf{A}$, the transpose of $\mathbf{A}$ and of $\mathbf{A}^{-1}$, respectively. In addition, $\underline{\mathbf{A}}^{(n)}$ denotes the mode-n unfolding matrix of $\boldsymbol{\mathcal{A}}$. Symbols "$\circledast$" and $\otimes$ are used to denote the Hadamard and Kronecker product. We denote by "$\times_n^1$" and "$\boxplus_n$" the tensor-train contraction and tensor concatenation along the n-th dimension respectively, and by `reshape(.)` the tensor reshaping operator. Also, $\|.\|_F$ and $\|.\|_2$ denote the Frobenius and the ℓ_2 norms.

2. Problem Statement

In this work, we consider the streaming tensor-train decomposition of an N-th order incomplete streaming tensor $\boldsymbol{\mathcal{X}}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times I_N^t}$ fixing all but the last time (temporal) dimension I_N^t. Particularly, $\boldsymbol{\mathcal{X}}_t$ is derived from appending the incoming stream $\boldsymbol{\mathcal{Y}}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times W}$ (with $W \geq 1$) to the last observation $\boldsymbol{\mathcal{X}}_{t-1}$ along the time dimension, i.e., $\boldsymbol{\mathcal{X}}_t = \boldsymbol{\mathcal{X}}_{t-1} \boxplus_N \boldsymbol{\mathcal{Y}}_t$ with $I_N^t = I_N^{t-1} + W$. We suppose that $\boldsymbol{\mathcal{Y}}_t$ is generated under the following model:

$$\boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{P}}_t \circledast \left(\boldsymbol{\mathcal{L}}_t + \boldsymbol{\mathcal{N}}_t\right). \tag{1}$$

Here, the binary mask $\boldsymbol{\mathcal{P}}_t$ indicates whether the $(i_1, i_2, \ldots, i_N)$-th entry of $\boldsymbol{\mathcal{Y}}_t$ is missing or observed (i.e., $p_{i_1 i_2 \ldots i_N} = 0$ if $y_{i_1 i_2 \ldots i_N}$ is missing and $p_{i_1 i_2 \ldots i_N} = 1$ otherwise), $\boldsymbol{\mathcal{N}}_t$ is a Gaussian noise tensor, and both tensors are of the same size with $\boldsymbol{\mathcal{Y}}_t$. The low-rank component $\boldsymbol{\mathcal{L}}_t$ has the form

$$\boldsymbol{\mathcal{L}}_t = \boldsymbol{\mathcal{G}}_t^{(1)} \times_2^1 \boldsymbol{\mathcal{G}}_t^{(2)} \times_3^1 \cdots \times_N^1 \mathbf{G}_t^{(N)}, \tag{2}$$

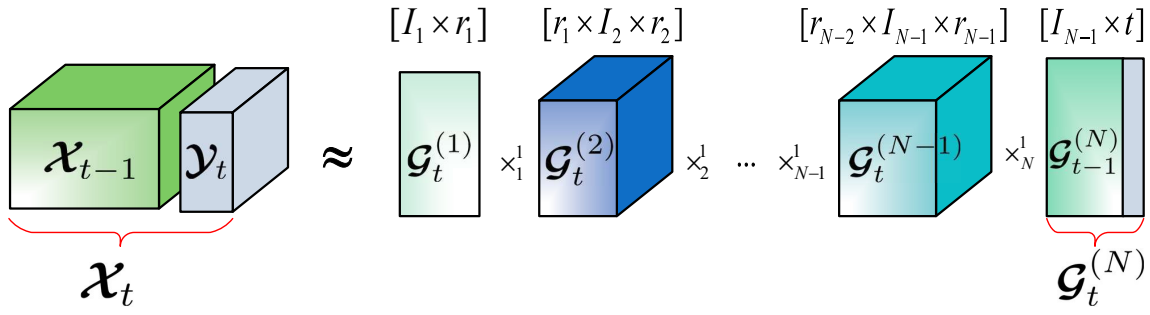


Figure 1: Streaming tensor-train decomposition of the streaming tensor $\mathcal{X}_t \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times I_N^t}$.

where $\mathcal{G}_t^{(n)} \in \mathbb{R}^{r_{n-1} \times I_n \times r_n}$ for $n = 1, 2, \ldots, N$ with $r_0 = r_N = 1$ is the n-th TT-core (the first and last TT-cores are indeed matrices); $[r_1, r_2, \ldots, r_{N-1}]$ is the TT-rank; and $\mathbf{G}_t^{(N)} \in \mathbb{R}^{r_{N-1} \times W}$ contains the last W columns of the temporal TT-core $\mathcal{G}_t^{(N)}$, i.e., $\mathcal{G}_t^{(N)} = \left[\mathcal{G}_{t-1}^{(N)} \;\; \mathbf{G}_t^{(N)}\right]$, please see Fig. 1 for an illustration.

In online setting, retaking the batch TT methods to factorize the underlying tensor $\mathcal{X}_t$ becomes inefficient due to inherent time-variation and non-stationarity of data streams as well as their high complexity in both computation and storage cost. Therefore, we aim to develop a low cost and effective tracker to estimate the TT-cores of $\mathcal{X}_t$ in time. Specifically, we propose to minimize the following exponentially weighted least-squares objective function:

$$\left\{\mathcal{G}_t^{(n)}\right\}_{n=1}^{N} = \underset{\{\mathcal{G}^{(n)}\}_{n=1}^{N}}{\operatorname{argmin}} \left[\sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \mathcal{P}_\tau \circledast \left(\mathcal{Y}_\tau - \mathcal{G}^{(1)} \times_2^1 \cdots \times_{N-1}^1 \mathcal{G}^{(N-1)} \times_N^1 \mathbf{G}_\tau^{(N)} \right) \right\|_F^2 + \rho \sum_{n=1}^{N-1} \left\| \mathcal{G}^{(n)} - \mathcal{G}_{t-1}^{(n)} \right\|_F^2 \right], \tag{3}$$

where $\beta \in (0, 1]$ is a forgetting factor aimed at reducing the effect of distant observations as well as facilitating the tracking process in dynamic environments; and ρ is a regularization parameter for controlling the time variation of TT-cores between two consecutive instances. $\{\mathcal{G}_{t-1}^{(n)}\}_{n=1}^{N-1}$ represent previous estimates of $\{\mathcal{G}_t^{(n)}\}_{n=1}^{N-1}$ and they serve as prior information for the optimization problem (3). To support our deployment in Section 3, we make two mild assumptions on the data model: TT-cores $\{\mathcal{G}_t^{(n)}\}_{n=1}^{N-1}$ may either be static or vary slowly with time, i.e., $\mathcal{G}_t^{(n)} \simeq \mathcal{G}_{t-1}^{(n)}$; and TT-rank is supposed to be known.

3. Proposed Method

In this section, we propose an adaptive method called ATT for adaptive tensor-train decomposition with missing data. Thanks to the block-coordinate descent (BCD) framework, we particularly decompose (3) into two main stages: first, update the temporal $\boldsymbol{\mathcal{G}}_t^{(N)}$ given old estimations $\{\boldsymbol{\mathcal{G}}_{t-1}^{(n)}\}_{n=1}^{N-1}$; and second, estimate the non-temporal $\boldsymbol{\mathcal{G}}_t^{(n)}$ given $\boldsymbol{\mathcal{G}}_t^{(N)}$ and remaining TT-cores, for $n = 1, 2, \ldots, N-1$. In stage 1, we apply the well-known regularized least-squares method for estimating $\boldsymbol{\mathcal{G}}_t^{(N)}$. An elegant recursive least-squares (RLS) adaptive filter is specifically developed to update the non-temporal TT-cores $\{\boldsymbol{\mathcal{G}}_t^{(n)}\}_{n=1}^{N-1}$ in an effective way. Main steps of the proposed ATT method are summarized in Algorithm 1.

3.1. ***Estimation of the temporal TT-core*** $\boldsymbol{\mathcal{G}}_t^{(N)}$

On the arrival of $\boldsymbol{\mathcal{Y}}_t$, we obtain $\mathbf{G}_t^{(N)}$ from

$$\mathbf{G}_t^{(N)} = \underset{\mathbf{G}^{(N)}}{\operatorname{argmin}} \left\| \boldsymbol{\mathcal{P}}_t \circledast \left(\boldsymbol{\mathcal{Y}}_t - \boldsymbol{\mathcal{H}}_{t-1} \times_N^1 \mathbf{G}^{(N)} \right) \right\|_F^2 + \lambda \left\| \mathbf{G}^{(N)} \right\|_F^2, \tag{4}$$

where $\boldsymbol{\mathcal{H}}_{t-1} = \boldsymbol{\mathcal{G}}_{t-1}^{(1)} \times_2^1 \boldsymbol{\mathcal{G}}_{t-1}^{(2)} \times_3^1 \cdots \times_{N-1}^1 \boldsymbol{\mathcal{G}}_{t-1}^{(N-1)}$ and $\lambda > 0$ is a small regularized parameter. Here, the first term of (4) is aimed at minimizing the residual error between observation and estimation for t-th temporal slice, while the introduction of $\lambda \| \mathbf{G}^{(N)} \|_F^2$ is for avoiding the ill-posed computation in practice. Particularly, we can rewrite (4) as follows

$$\mathbf{G}_t^{(N)} = \underset{\mathbf{G}^{(N)}}{\operatorname{argmin}} \left\| \mathbf{P}_t \circledast \left(\mathbf{Y}_t - \mathbf{H}_{t-1} \mathbf{G}^{(N)} \right) \right\|_2^2 + \lambda \left\| \mathbf{G}^{(N)} \right\|_F^2, \tag{5}$$

where $\mathbf{Y}_t, \mathbf{P}_t \in \mathbb{R}^{I_1 \ldots I_{N-1} \times W}$, and $\mathbf{H}_{t-1} \in \mathbb{R}^{I_1 \ldots I_{N-1} \times r_{N-1}}$ are the unfolding matrices of $\boldsymbol{\mathcal{Y}}_t$, $\boldsymbol{\mathcal{P}}_t$ and $\boldsymbol{\mathcal{H}}_{t-1}$, respectively. Furthermore, (5) can be decomposed into W subproblems w.r.t. W columns of $\mathbf{G}^{(N)}$:

$$\mathbf{G}_t^{(N)}(:, i) = \underset{\mathbf{g}_i}{\operatorname{argmin}} \left\| \overline{\mathbf{P}}_{t,i} \left(\mathbf{y}_{t,i} - \mathbf{H}_{t-1} \mathbf{g}_i \right) \right\|_2^2 + \lambda \left\| \mathbf{g}_i \right\|_2^2. \tag{6}$$

where $\mathbf{y}_{t,i} = \mathbf{Y}_t(:, i)$ and $\overline{\mathbf{P}}_{t,i} = \mathtt{diag}\{\mathbf{P}_t(:, i)\}$. The closed-form solution of the regularized least-squares (6) can be given by

$$\mathbf{G}_t^{(N)}(:, i) = \left(\mathbf{H}_{t-1}^\top \overline{\mathbf{P}}_{t,i} \mathbf{H}_{t-1} + \lambda \mathbf{I}_{r_{N-1}} \right)^{-1} \mathbf{H}_{t-1}^\top \overline{\mathbf{P}}_{t,i} \mathbf{y}_{t,i}. \tag{7}$$

Then, the temporal TT-core $\boldsymbol{\mathcal{G}}_t^{(N)}$ is simply updated as $\boldsymbol{\mathcal{G}}_t^{(N)} = \left[\boldsymbol{\mathcal{G}}_{t-1}^{(N)} \;\; \mathbf{G}_t^{(N)} \right]$. Note that, we can re-update $\boldsymbol{\mathcal{G}}_t^{(N)}$ in the same way above when other TT-cores $\{\boldsymbol{\mathcal{G}}_t^{(n)}\}_{n=1}^{N-1}$ are updated.

Algorithm 1: ATT - Adaptive Tensor-Train Decomposition

Input:

+ Streams $\{\mathcal{P}_t \circledast \mathcal{Y}_t\}_{t=1}^{\infty}$, $\mathcal{P}_t, \mathcal{Y}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times W}$, TT-rank $\mathbf{r}_{\text{TT}} = [r_1, r_2, \ldots, r_{N-1}]$,
+ Forgetting factor $0 < \beta \leq 1$, regularized parameters $\rho, \lambda > 0$.

Output: TT-cores $\{\mathcal{G}_t^{(n)}\}_{n=1}^{N}$.

Initialization:

+ $\{\mathcal{G}_0^{(n)}\}_{n=1}^{N-1}$ are initialized at random,
+ $\{\mathbf{S}_0^{(n)}\}_{n=1}^{N-1} = \mathbf{0}$ and $\{\Delta\mathcal{G}_0^{(n)}\}_{n=1}^{N-1} = \mathbf{0}$.

for $t = 1, 2, \ldots$ **do**

Stage 1: Estimate the temporal TT-core $\mathcal{G}_t^{(N)}$

$\mathcal{H}_{t-1} = \mathcal{G}_{t-1}^{(1)} \times_2^1 \cdots \times_{N-1}^1 \mathcal{G}_{t-1}^{(N-1)}$

$\mathbf{H}_{t-1} = \texttt{reshape}\big\{\mathcal{H}_{t-1}, [I_1 I_2 \ldots I_{N-1}, r_{N-1}]\big\}$

for $i = 1, 2, \ldots, W$ **do**

$\mathbf{y}_{t,i} = \texttt{vec}\big\{\mathcal{Y}_t(:, \ldots, :, i)\big\}$

$\overline{\mathbf{P}}_{t,i} = \texttt{diag}\big\{\mathcal{P}_t(:, \ldots, :, i)\big\}$

$\mathbf{G}_t^{(N)}(:, i) = \big(\mathbf{H}_{t-1}^\top \overline{\mathbf{P}}_{t,i} \mathbf{H}_{t-1} + \lambda \mathbf{I}_{r_{N-1}}\big)^{-1} \mathbf{H}_{t-1}^\top \overline{\mathbf{P}}_{t,i} \mathbf{y}_{t,i}$

$\delta\mathbf{y}_{t,i} = \overline{\mathbf{P}}_{t,i}\big(\mathbf{y}_{t,i} - \mathbf{H}_{t-1}\mathbf{G}_t^{(N)}(:, i)\big)$

$\Delta\mathcal{Y}_{t,i} = \texttt{reshape}\big\{\delta\mathbf{y}_{t,i}, [I_1, I_2, \ldots, I_{N-1}, 1]\big\}$

end

$\mathcal{G}_t^{(N)} = \big[\mathcal{G}_{t-1}^{(N)} \;\; \mathbf{G}_t^{(N)}\big]$

$\Delta\mathcal{Y}_t = \Delta\mathcal{Y}_{t,1} \boxplus_N \Delta\mathcal{Y}_{t,2} \boxplus_N \cdots \boxplus_N \Delta\mathcal{Y}_{t,W}$

Stage 2: Estimate the non-temporal TT-cores $\{\mathcal{G}_t^{(n)}\}_{n=1}^{N-1}$

for $n = 1, 2, \ldots, N-1$ **do**

$\mathcal{A}_{t-1}^{(n)} = \mathcal{G}_{t-1}^{(1)} \times_2^1 \cdots \times_{n-1}^1 \mathcal{G}_{t-1}^{(n-1)}$

$\mathbf{A}_{t-1}^{(n)} = \texttt{reshape}\big\{\mathcal{A}_{t-1}^{(n)}, [r_{n-1}, I_1 I_2 \ldots I_{n-1}]\big\}$

$\mathcal{B}_t^{(n)} = \mathcal{G}_{t-1}^{(n+1)} \times_{n+2}^1 \cdots \mathcal{G}_{t-1}^{(N-1)} \times_N^1 \mathbf{G}_t^{(N)}$

$\mathbf{B}_t^{(n)} = \texttt{reshape}\big\{\mathcal{B}_t^{(n)}, [r_n, I_{n+1} \ldots I_{N-1}]\big\}$

$\mathbf{W}_t^{(n)} = \mathbf{B}_t^{(n)} \otimes \mathbf{A}_{t-1}^{(n)}$

$\mathbf{S}_t^{(n)} = \beta\mathbf{S}_{t-1}^{(n)} + \mathbf{W}_t^{(n)}(\mathbf{W}_t^{(n)})^\top$

$\Delta\underline{\mathbf{G}}_t^{(n)} = \Big(\big(\underline{\mathbf{P}}_t^{(n)} \otimes \Delta\underline{\mathbf{Y}}_t^{(n)}\big)\big(\mathbf{W}_t^{(n)}\big)^\top + \beta\rho\Delta\underline{\mathbf{G}}_{t-1}^{(n)}\Big)\Big(\mathbf{S}_t^{(n)} + \rho\mathbf{I}_{r_{n-1}r_n}\Big)^{-\top}$

$\underline{\mathbf{G}}_t^{(n)} = \underline{\mathbf{G}}_{t-1}^{(n)} + \Delta\underline{\mathbf{G}}_t^{(n)}$

$\mathcal{G}_t^{(n)} = \texttt{reshape}\big\{\underline{\mathbf{G}}_t^{(n)}, [r_{n-1}, I_n, r_n]\big\}$

end

Stage 3 (Optional): Re-estimate $\mathcal{G}_t^{(N)}$ with updated $\{\mathcal{G}_t^{(n)}\}_{n=1}^{N-1}$ as in Stage 1.

end

3.2. *Estimation of the non-temporal TT-cores* $\{\mathcal{G}_t^{(n)}\}_{n=1}^{N-1}$

We update each $\mathcal{G}_t^{(n)}$ by minimizing

$$\mathcal{G}_t^{(n)} = \underset{\mathcal{G}^{(n)}}{\operatorname{argmin}} \left[\sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \mathcal{P}_\tau \circledast \left(\mathcal{Y}_\tau - \mathcal{A}_{t-1}^{(n)} \times_n^1 \mathcal{G}^{(n)} \times_{n+1}^1 \mathcal{B}_\tau^{(n)} \right) \right\|_F^2 + \rho \left\| \mathcal{G}^{(n)} - \mathcal{G}_{t-1}^{(n)} \right\|_F^2 \right], \tag{8}$$

where $\mathcal{A}_{t-1}^{(n)} = \mathcal{G}_{t-1}^{(1)} \times_2^1 \mathcal{G}_{t-1}^{(2)} \times_3^1 \cdots \times_{n-1}^1 \mathcal{G}_{t-1}^{(n-1)}$ and $\mathcal{B}_\tau^{(n)} = \mathcal{G}_{t-1}^{(n+1)} \times_{n+2}^1 \cdots \times_{N-1}^1 \mathcal{G}_{t-1}^{(N-1)} \times_N^1 \mathbf{G}_\tau^{(N)}$. We further recast (8) as

$$\mathbf{G}_t^{(n)} = \underset{\mathbf{G}^{(n)}}{\operatorname{argmin}} \left[\sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \underline{\mathbf{P}}_\tau^{(n)} \circledast \left(\underline{\mathbf{Y}}_\tau^{(n)} - \mathbf{G}^{(n)} \mathbf{W}_\tau^{(n)} \right) \right\|_F^2 + \rho \left\| \mathbf{G}^{(n)} - \mathbf{G}_{t-1}^{(n)} \right\|_F^2 \right], \tag{9}$$

where $\mathbf{G}_t^{(n)} = \texttt{reshape}\{\mathcal{G}_t^{(n)}, [I_n, r_{n-1} r_n]\}$; $\underline{\mathbf{P}}_\tau^{(n)}$ and $\underline{\mathbf{Y}}_\tau^{(n)}$ are the mode-n unfolding matrices of $\mathcal{P}_\tau$ and $\mathcal{Y}_\tau$, respectively; $\mathbf{W}_\tau^{(n)} = \mathbf{B}_\tau^{(n)} \otimes \mathbf{A}_{t-1}^{(n)}$ where $\mathbf{A}_{t-1}^{(n)} = \texttt{reshape}\{\mathcal{A}_{t-1}^{(n)}, [r_{n-1}, I_1 \ldots I_{n-1}]\}$ and $\mathbf{B}_\tau^{(n)} = \texttt{reshape}\{\mathcal{B}_t^{(n)}, [r_n, I_{n+1} \ldots I_{N-1}]\}$.

Similar to the update of $\mathbf{G}_t^{(N)}$ in the first stage, we can update independently each row $\mathbf{g}_{t,m}^{(n)}$ of $\mathbf{G}_t^{(n)}$ (with $m = 1, 2, \ldots, I_n$) as follows:

$$\mathbf{g}_{t,m}^{(n)} = \underset{\mathbf{g}_m^{(n)}}{\operatorname{argmin}} \left[\sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \overline{\mathbf{P}}_{\tau,m}^{(n)} \left(\mathbf{y}_{\tau,m}^{(n)} - \mathbf{g}_m^{(n)} \mathbf{W}_\tau^{(n)} \right)^\top \right\|_2^2 + \rho \left\| \mathbf{g}_m^{(n)} - \mathbf{g}_{t-1,m}^{(n)} \right\|_2^2 \right], \tag{10}$$

where $\mathbf{y}_{\tau,m}^{(n)} = \underline{\mathbf{Y}}_\tau^{(n)}(m, :)$ and $\overline{\mathbf{P}}_{\tau,m}^{(n)} = \texttt{diag}\{\underline{\mathbf{P}}_\tau^{(n)}(m, :)\}$. Specifically, $\mathbf{g}_{t,m}^{(n)}$ can be derived from setting the gradient of the function in (10) to zero:

$$\left(\rho \mathbf{I}_{r_{n-1} r_n} + \sum_{\tau=1}^{t} \beta^{t-\tau} \mathbf{W}_\tau^{(n)} \overline{\mathbf{P}}_{\tau,m}^{(n)} \left(\mathbf{W}_\tau^{(n)} \right)^\top \right) \left(\mathbf{g}_m^{(n)} \right)^\top = \rho \left(\mathbf{g}_{t-1,m}^{(n)} \right)^\top + \sum_{\tau=1}^{t} \beta^{t-\tau} \mathbf{W}_\tau^{(n)} \overline{\mathbf{P}}_{\tau,m}^{(n)} \left(\mathbf{y}_{\tau,m}^{(n)} \right)^\top. \tag{11}$$

The closed-form solution of (11) is then given by

$$\mathbf{g}_{t,m}^{(n)} = \left[\left(\mathbf{S}_{t,m}^{(n)} + \rho \mathbf{I}_{r_{n-1} r_n} \right)^{-1} \left(\mathbf{d}_{t,m}^{(n)} + \rho \left(\mathbf{g}_{t-1,m}^{(n)} \right)^\top \right) \right]^\top, \tag{12}$$

where $\mathbf{S}_{t,m}^{(n)}$ and $\mathbf{d}_{t,m}^{(n)}$ can be recursively updated as $\mathbf{S}_{t,m}^{(n)} = \beta \mathbf{S}_{t-1,m}^{(n)} + \mathbf{W}_t^{(n)} \overline{\mathbf{P}}_{t,m}^{(n)} \left(\mathbf{W}_t^{(n)} \right)^\top$ and $\mathbf{d}_{t,m}^{(n)} = \beta \mathbf{d}_{t-1,m}^{(n)} + \mathbf{W}_t^{(n)} \overline{\mathbf{P}}_{t,m}^{(n)} \left(\mathbf{y}_{t,m}^{(n)} \right)^\top$. Here, the two auxiliary variables, $\mathbf{S}_{t,m}^{(n)}$ and $\mathbf{d}_{t,m}^{(n)}$, represent two weighted summations of products $\left\{ \mathbf{W}_\tau^{(n)} \overline{\mathbf{P}}_{\tau,m}^{(n)} \left(\mathbf{W}_\tau^{(n)} \right)^\top \right\}_{\tau=1}^{t}$ and $\left\{ \mathbf{W}_\tau^{(n)} \overline{\mathbf{P}}_{\tau,m}^{(n)} \left(\mathbf{y}_{\tau,m}^{(n)} \right)^\top \right\}_{\tau=1}^{t}$, respectively. At time t, each matrix $\mathbf{W}_\tau^{(n)}$ is constructed using the most recent estimates of TT-cores $\{\mathcal{G}_{t-1}^{(n)}\}_{n=1}^{N-1}$ instead of $\{\mathcal{G}_\tau^{(n)}\}_{n=1}^{N-1}$ as in the classical recursive least-squares (RLS) method. This serves as an approximation intended to save computations by avoiding the retrieval of old information of TT-cores at distant lags. Moreover, this approach enables an

elegant transformation of the closed-form solution (12) into a recursive one. To be specific, we can represent (12) as follows

$$
\begin{aligned}
\big(\mathbf{g}_{t,m}^{(n)}\big)^{\top} &= \Big(\mathbf{S}_{t,m}^{(n)} + \rho\mathbf{I}_{r_{n-1}r_n}\Big)^{-1}\Big(\beta\mathbf{d}_{t-1,m}^{(n)} + \mathbf{W}_t^{(n)}\overline{\mathbf{P}}_{t,m}^{(n)}\big(\mathbf{x}_{t,m}^{(n)}\big)^{\top} + \rho\big(\mathbf{g}_{t-1,m}^{(n)}\big)^{\top}\Big) \\
&= \Big(\mathbf{S}_{t,m}^{(n)} + \rho\mathbf{I}_{r_{n-1}r_n}\Big)^{-1}\Big[\beta\Big(\big(\mathbf{S}_{t-1,m}^{(n)} + \rho\mathbf{I}_{r_{n-1}r_n}\big)\big(\mathbf{g}_{t-1,m}^{(n)}\big)^{\top} - \rho\big(\mathbf{g}_{t-2,m}^{(n)}\big)^{\top}\Big) + \mathbf{W}_t^{(n)}\overline{\mathbf{P}}_{t,m}^{(n)}\big(\mathbf{x}_{t,m}^{(n)}\big)^{\top} + \rho\big(\mathbf{g}_{t-1,m}^{(n)}\big)^{\top}\Big] \\
&= \Big(\mathbf{S}_{t,m}^{(n)} + \rho\mathbf{I}_{r_{n-1}r_n}\Big)^{-1}\Big[\Big(\underbrace{\beta\mathbf{S}_{t-1,m}^{(n)} + \mathbf{W}_t^{(n)}\overline{\mathbf{P}}_{t,m}^{(n)}\big(\mathbf{W}_t^{(n)}\big)^{\top} + \rho\mathbf{I}_{r_{n-1}r_n}}_{=\ \mathbf{S}_{t,m}^{(n)} + \rho\mathbf{I}_{r_{n-1}r_n}}\Big)\big(\mathbf{g}_{t-1,m}^{(n)}\big)^{\top} + \beta\rho\Big(\mathbf{g}_{t-1,m}^{(n)} - \mathbf{g}_{t-2,m}^{(n)}\Big)^{\top} \\
&\qquad + \mathbf{W}_t^{(n)}\overline{\mathbf{P}}_{t,m}^{(n)}\Big(\mathbf{x}_{t,m}^{(n)} - \mathbf{g}_{t-1,m}^{(n)}\mathbf{W}_t^{(n)}\Big)^{\top}\Big] \\
&= \big(\mathbf{g}_{t-1,m}^{(n)}\big)^{\top} + \Big(\mathbf{S}_{t,m}^{(n)} + \rho\mathbf{I}_{r_{n-1}r_n}\Big)^{-1}\Big[\beta\rho\Big(\mathbf{g}_{t-1,m}^{(n)} - \mathbf{g}_{t-2,m}^{(n)}\Big)^{\top} + \mathbf{W}_t^{(n)}\overline{\mathbf{P}}_{t,m}^{(n)}\Big(\mathbf{x}_{t,m}^{(n)} - \mathbf{g}_{t-1,m}^{(n)}\mathbf{W}_t^{(n)}\Big)^{\top}\Big].
\end{aligned}
\tag{13}
$$

As a result, we derive the following recursive update

$$
\mathbf{g}_{t,m}^{(n)} = \mathbf{g}_{t-1,m}^{(n)} + \Big(\boldsymbol{\delta}\mathbf{y}_{t,m}^{(n)}\overline{\mathbf{P}}_{t,m}^{(n)}\big(\mathbf{W}_t^{(n)}\big)^{\top} + \beta\rho\boldsymbol{\delta}\mathbf{g}_{t-1,m}^{(n)}\Big)\Big(\mathbf{S}_{t,m}^{(n)} + \rho\mathbf{I}_{r_{n-1}r_n}\Big)^{-\top}, \tag{14}
$$

where $\boldsymbol{\delta}\mathbf{y}_{t,m}^{(n)} = \overline{\mathbf{P}}_{t,m}^{(n)}\big(\mathbf{y}_{t,m}^{(n)} - \mathbf{g}_{t-1,m}^{(n)}\mathbf{W}_t^{(n)}\big)^{\top}$ and $\boldsymbol{\delta}\mathbf{g}_{t-1,m}^{(n)} = \mathbf{g}_{t-1,m}^{(n)} - \mathbf{g}_{t-2,m}^{(n)}$.

To enhance the computational efficiency, we can further simplify $\mathbf{S}_{t,m}^{(n)}$ in (14) by excluding the diagonal matrix $\overline{\mathbf{P}}_{t,m}^{(n)}$ as follows

$$
\mathbf{S}_{t,m}^{(n)} \approx \beta\mathbf{S}_{t,m}^{(n)} + \mathbf{W}_t^{(n)}\big(\mathbf{W}_t^{(n)}\big)^{\top}. \tag{15}
$$

Accordingly, we can set a shared auxiliary matrix $\mathbf{S}_t^{(n)}$ for every row $\{\mathbf{g}_{t,m}^{(n)}\}_{m=1}^{I_n}$ of $\mathbf{G}_t^{(n)}$ as

$$
\mathbf{S}_t^{(n)} = \beta\mathbf{S}_t^{(n)} + \mathbf{W}_t^{(n)}\big(\mathbf{W}_t^{(n)}\big)^{\top}. \tag{16}
$$

Then, a recursive rule with a lower space complexity for updating the whole matrix $\mathbf{G}_t^{(n)}$ at the same time is given by

$$
\mathbf{G}_t^{(n)} = \mathbf{G}_{t-1}^{(n)} + \Big(\big(\underline{\mathbf{P}}_t^{(n)} \circledast \boldsymbol{\Delta}\underline{\mathbf{Y}}_t^{(n)}\big)\big(\mathbf{W}_t^{(n)}\big)^{\top} + \beta\rho\boldsymbol{\Delta}\mathbf{G}_{t-1}^{(n)}\Big)\Big(\mathbf{S}_t^{(n)} + \rho\mathbf{I}_{r_{n-1}r_n}\Big)^{-\top}, \tag{17}
$$

where $\boldsymbol{\Delta}\underline{\mathbf{Y}}_{t,m}^{(n)} = \underline{\mathbf{Y}}_t^{(n)} - \mathbf{G}_{t-1}^{(n)}\mathbf{W}_t^{(n)}$ and $\boldsymbol{\Delta}\mathbf{G}_{t-1}^{(n)} = \mathbf{G}_{t-1}^{(n)} - \mathbf{G}_{t-2}^{(n)}$. We then set $\boldsymbol{\mathcal{G}}_t^{(n)} = \texttt{reshape}\big\{\mathbf{G}_t^{(n)}, [r_{n-1}, I_n, r_n]\big\}$.

By following the aforementioned framework, the proposed ATT tracker can be considered as an "indirect" recursive least-squares (RLS) algorithm. Instead of directly applying the classical RLS method to minimize the primary optimization (9), we leverage the insight that

solving the main objective function, which is represented by the exponentially weighted least-squares function in (9), can be simplified to minimizing sub-problems for each row of the TT core. These sub-problems involve the use of recursive procedures and the implementation of approximations, resulting in a reduction in both computational complexity and memory storage. The RLS solution of (9) is subsequently obtained by consolidating the recursive solutions acquired from these sub-problems. The update rule (17) also reveals that ATT can support parallel and distributed computing. It stems from the fact that all auxiliary matrices $\boldsymbol{\Delta}\underline{\mathbf{Y}}_t$, $\mathbf{W}_t^{(n)}$, $\boldsymbol{\Delta}\mathbf{G}_{t-1}^{(n)}$, and $\mathbf{S}_t^{(n)}$ for updating $\boldsymbol{\mathcal{G}}_t^{(n)}$ are independent of $\{\boldsymbol{\mathcal{G}}_t^{(m)}\}_{m\neq n}^{N-1}$. Therefore, we can assign $N-1$ individual computers to update $\{\boldsymbol{\mathcal{G}}_t^{(n)}\}_{n=1}^{N-1}$ in parallel without disrupting the remaining TT-cores. In other words, TT-cores can be updated simultaneously on the arrival of new data at each time t.

3.3. ***Performance Analysis***

For brevity, we assume that $I_n = I$ and $r_n = r$ for all $n = 1, 2, \ldots, N-1$. At time t, ATT requires a cost of $\boldsymbol{\mathcal{O}}(W|\Omega_t|r^2)$ flops for updating $\mathbf{G}_t^{(N)}$ where $|\Omega_t|$ denotes the number of observed data. Most of operations for updating $\boldsymbol{\mathcal{G}}_t^{(n)}$ are matrix-matrix products except an inverse operation of a $r^2 \times r^2$ matrix. Thus, ATT requires an extra cost of $\boldsymbol{\mathcal{O}}\big((N-1)I^{N-1}r^4\big)$ flops. The overall complexity of ATT is $\boldsymbol{\mathcal{O}}\big(r^2 \max\big\{(N-1)I^{N-1}r^2, W|\Omega_t|\big\}\big)$ flops. In term of memory storage, ATT needs $\boldsymbol{\mathcal{O}}\big((N-1)(2Ir^2+r^4)\big)$ words of memory for storing $\big\{\boldsymbol{\mathcal{G}}_t^{(n)}\big\}_{n=1}^{N-1}$, $\big\{\boldsymbol{\Delta}\boldsymbol{\mathcal{G}}_t^{(n)}\big\}_{n=1}^{N-1}$, and $\big\{\mathbf{S}_t^{(n)}\big\}_{n=1}^{N-1}$.

Compared to batch TT methods (e.g., TT-SVD [2] and TT-HSVD [22]), the cost of ATT is much cheaper as it is independent of the temporal dimension. Besides, its computation involves only cheap matrix-matrix products and inverse operations of small matrices, and hence, it avoids the expensive computation of SVD on the tensor's unfolding matrices. Compared to TT-FOA that is the first and only adaptive algorithm for streaming TT decomposition in the literature, ATT shares the same computational and space complexity.

4. Experiments

In this section, we investigate the tracking ability of ATT with respect to the following aspects: additive noise effect, and its performance in nonstationary environments. Its effectiveness for real data is demonstrated with the problem of online video completion in comparison with the state-of-the-art tensor tracking algorithms.[2]

[2]Our MATLAB codes are available online at: `https://github.com/thanhtbt/ATT-miss/`.

Experiment Setup: At time t, the t-th incomplete slice $\boldsymbol{\mathcal{Y}}_t$ is generated at random under the following model:

$$\boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{P}}_t \circledast \left(\boldsymbol{\mathcal{G}}_t^{(1)} \times_2^1 \boldsymbol{\mathcal{G}}_t^{(2)} \times_3^1 \boldsymbol{\mathcal{G}}_t^{(3)} \times_4^1 \mathbf{g}_t^{(4)} + \boldsymbol{\mathcal{N}}_t\right). \tag{18}$$

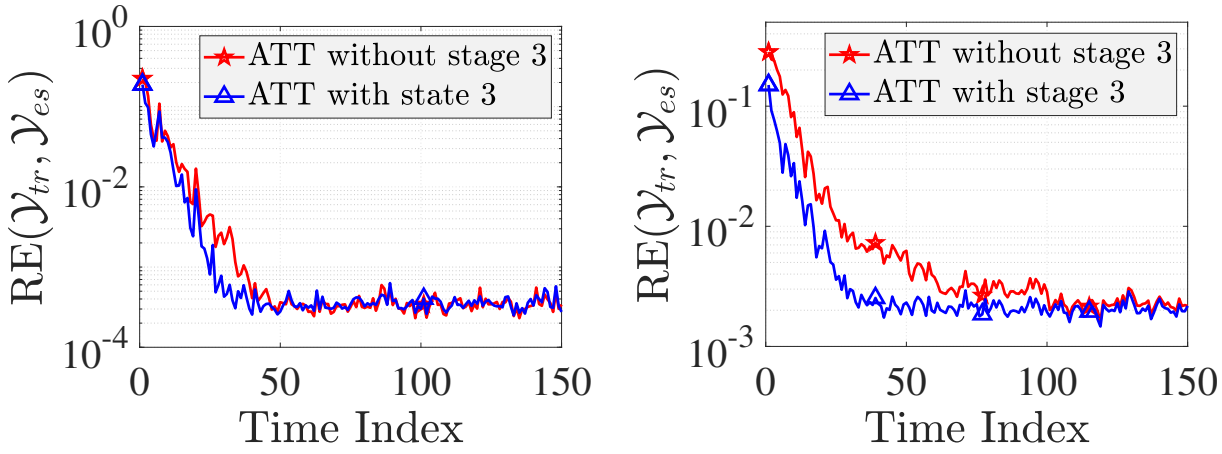
Here, $\boldsymbol{\mathcal{P}}_t \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times 1}$ is a binary tensor whose entries are i.i.d. Bernoulli random variables with probability $1 - \omega_{\texttt{miss}}$, i.e., $\omega_{\texttt{miss}}$ represents the missing density of $\boldsymbol{\mathcal{Y}}_t$. Entries of the noise tensor $\boldsymbol{\mathcal{N}}_t$ are i.i.d. from $\mathcal{N}(0, \sigma_n^2)$. $\mathbf{g}_t^{(4)} \in \mathbb{R}^{r_3 \times 1}$ is a Gaussian vector of zero-mean and unit-variance. TT-cores $\boldsymbol{\mathcal{G}}_t^{(1)}, \boldsymbol{\mathcal{G}}_t^{(2)}$, and $\boldsymbol{\mathcal{G}}_t^{(3)}$ are of size $I_1 \times r_1$, $r_1 \times I_2 \times r_2$, and $r_2 \times I_3 \times r_3$, respectively. Their time variation is modelled as follows $\boldsymbol{\mathcal{G}}_t^{(n)} = \boldsymbol{\mathcal{G}}_{t-1}^{(n)} + \varepsilon \boldsymbol{\mathcal{V}}_t^{(n)}$, for $n = 1, 2, 3$, where ε plays a role as the time-varying factor, $\boldsymbol{\mathcal{V}}_t^{(n)}$ is of the same size as $\boldsymbol{\mathcal{G}}_t^{(n)}$ and its entries are also i.i.d from $\mathcal{N}(0, 1)$. We use the following relative error (RE) metric to evaluate the estimation accuracy:

$$\mathrm{RE}\left(\boldsymbol{\mathcal{Y}}_{tr}, \boldsymbol{\mathcal{Y}}_{es}\right) = \left\|\boldsymbol{\mathcal{Y}}_{tr} - \boldsymbol{\mathcal{Y}}_{es}\right\|_F \Big/ \left\|\boldsymbol{\mathcal{Y}}_{tr}\right\|_F, \tag{19}$$

where $\boldsymbol{\mathcal{Y}}_{tr}$ (resp. $\boldsymbol{\mathcal{Y}}_{es}$) refers to the true tensor (resp. reconstructed tensor). In all experiments, we use ATT with only two key stages I and II. Indeed, the two versions of ATT (with and without stage III) exhibit the same estimation accuracy when the number of observations is large, as illustrated in Fig. 2. In a time-varying environment, the inclusion of stage III can improve the performance of ATT in terms of both both estimation accuracy and convergence rate at the early stage of tracking process, as shown in Fig. 2(b).

Effect of the noise level σ_n: We vary the value of σ_n and evaluate the performance of ATT. Here, we used a static tensor (i.e., $\varepsilon = 0$) of size $20 \times 20 \times 20 \times 1000$ and rank $\mathbf{r}_{\mathrm{TT}} = [5, 5, 5]$. The missing density $\omega_{\texttt{miss}}$ was set to 10%. We fixed the forgetting factor β and the two regularized parameters ρ, λ, at 0.5, 1, and 1, respectively. A significant change was also created at $t = 600$ (i.e., we set $\varepsilon = 1$ when $t = 600$ and $\varepsilon = 0$ otherwise) to investigate how fast ATT could converge. The result is illustrated in Fig. 3. We can see that the noise level σ_n does not affect the convergence rate of ATT but only its estimation error.

Effect of the time-varying factor ε: We next investigate the tracking ability of ATT in nonstationary environments. Similar to the previous experiment, we also vary the value of ε and then evaluate its estimation accuracy. Most of experimental parameters were kept as above, except the noise level σ_n which was set to 10^{-3}. Fig. 4 illustrates the performance of ATT versus the value of ε. We can see that the estimation accuracy of ATT goes down when ε increases, but converges towards a steady-state error in the similar manner as in



(a) Stationary environment with the time-varying factor $\varepsilon = 0$ and 10% missing data.

(b) Nonstationary environment with the time-varying factor $\varepsilon = 10^{-2}$ and 90% missing data.

Figure 2: Performance comparison between two versions of ATT on an incomplete streaming tensor of size $10 \times 15 \times 20 \times 150$ and rank $r_1 = r_2 = r_3 = 5$ with the noise level $\sigma_n = 10^{-3}$.

the previous case. Intuitively, the time-varying factor has an influence on the tracking performance of recursive least-squares (RLS) methods.[3] However, as shown in Fig. 4, the value of ε does not affect ATT's convergence rate. This "phenomenon" thus deserves further investigations.

Effect of the missing density $\omega_{\texttt{miss}}$: Here, we measure the performance of ATT in the presence of different missing densities. Particularly, the value of $\omega_{\texttt{miss}}$ was chosen among $\{20\%, 40\%, 80\%\}$. We reused the same 4-order streaming tensor above with $\sigma_n = \varepsilon = 10^{-3}$. Fig. 5 shows that the number of missing entries in $\boldsymbol{\mathcal{X}}_t$ has an impact on both convergence rate and estimation accuracy of ATT, i.e., the lower the value of $\omega_{\texttt{miss}}$ is, the better performance ATT achieves. However, even with 80% missing data, ATT is still able to achieve relatively good performance.

ATT vs the state-of-the-art algorithms: In this task, we compare ATT with TT-FOA and its stochastic variant TT-FOA-S [14]. We fixed the forgetting factor λ of TT-FOA at 0.5. As the conventional TT-FOA and TT-FOA-S are not designed for missing data,

[3]It is very well known that one of main sources of the tracking error is due to the time-varying signals [25]. Particularly, this error called "lag-error" is caused by the attempt of adaptive RLS methods to track the variation of data over time. Here, the time-varying factor is used to model/generate such nonstationary signals, and hence, it is expected to have an impact on the convergence behavior of ATT.

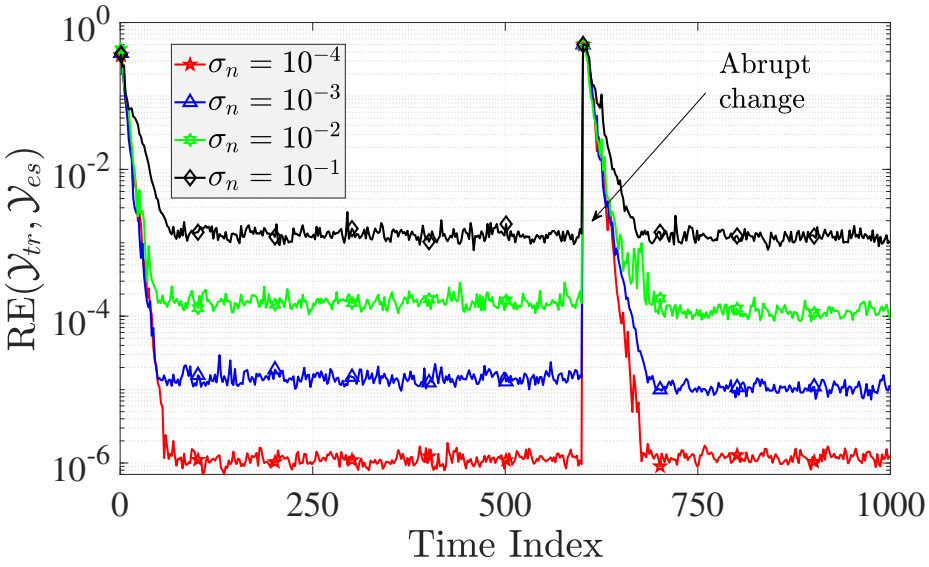


Figure 3: Effect of the noise level σ_n on the tracking ability of ATT.

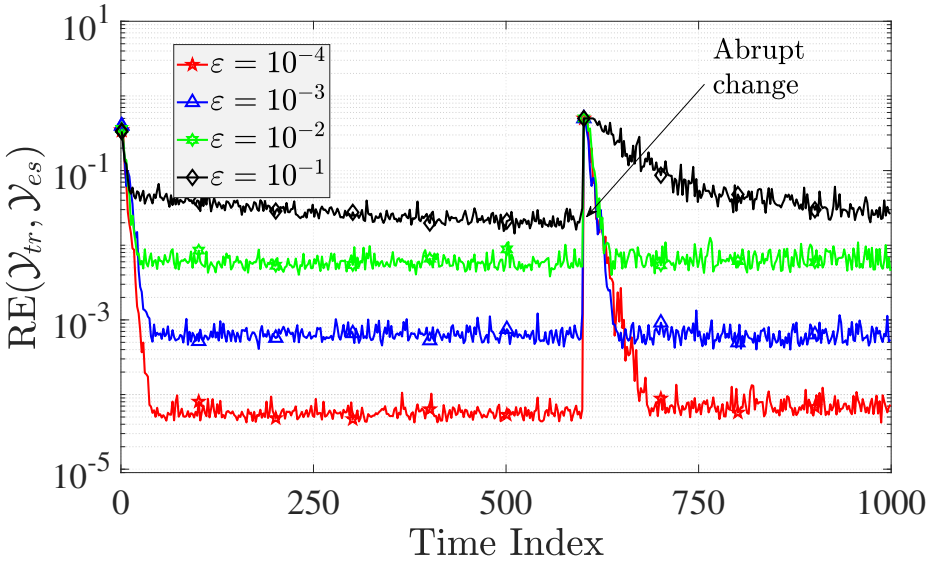


Figure 4: Effect of the time-varying factor ε on the tracking ability of ATT.

we reformulated their update rule by simply putting an observation mask on the data. We reused the same experiment setup as in the previous tasks. Performance comparison results are illustrated in Fig. 6 and Fig. 7. We can see that in the presence of full observations, three algorithms provide the similar estimation accuracy. However, the convergence rate of TT-FOA-S is slower than that of ATT and TT-FOA. In the presence of missing data, ATT

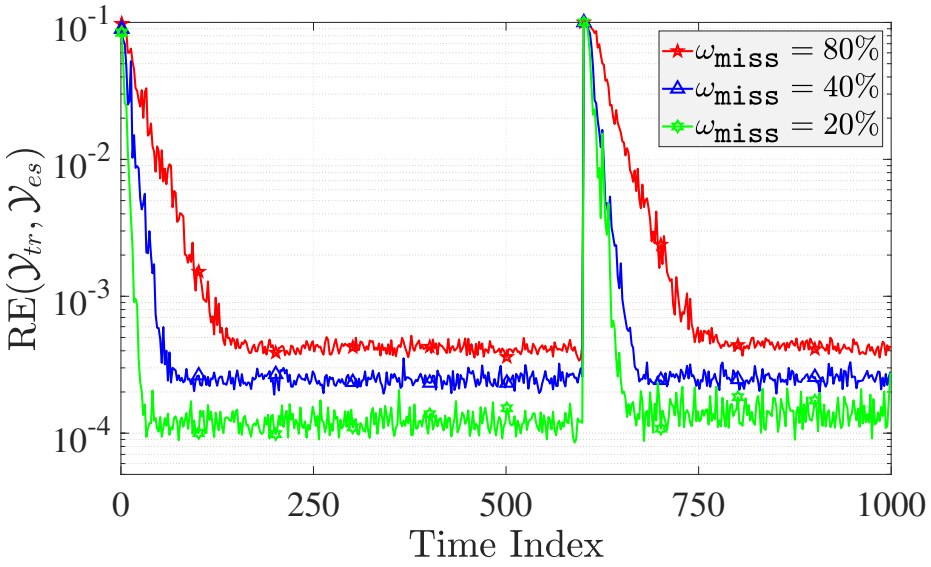
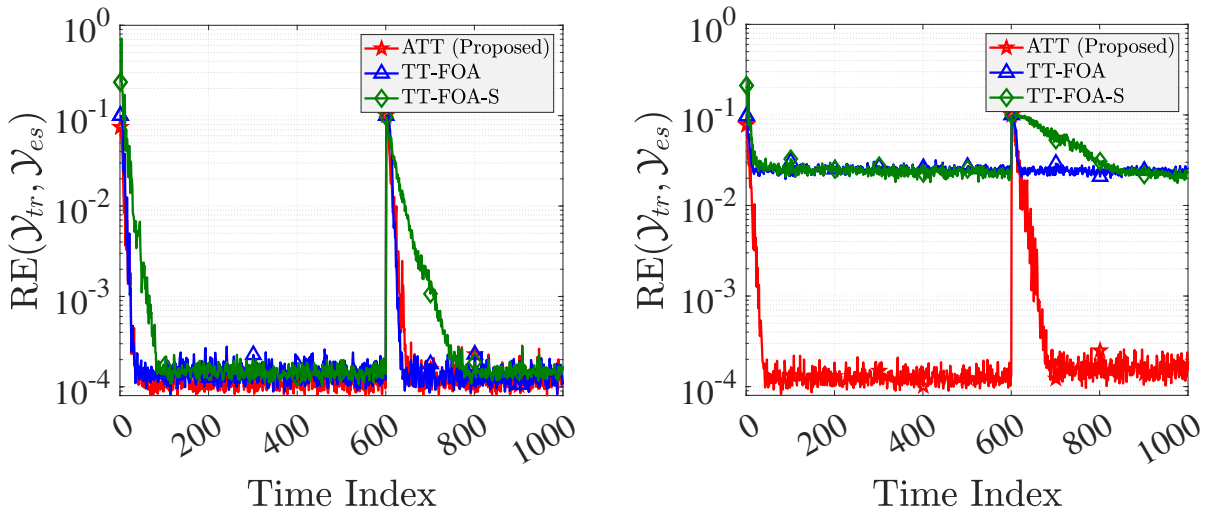


Figure 5: Effect of the missing density $\omega_{\texttt{miss}}$ on the tracking ability of ATT.

outperforms others. The use of a binary observation mask inadvertently introduces "noise" into the data, as the zero entries do not accurately represent the true values of the missing entries. Consequently, both TT-FOA and TT-FOA-S are subjected to additional additive noise, alongside the standard Gaussian noise. It explains why TT-FOA and TT-FOA-S fail to track the underlying tensor in the presence of missing data. In terms of run time, ATT demonstrates comparable performance to state-of-the-art adaptive TT methods.

Online video completion: Three real video sequences are used in this task, including "Lobby", "Highway", and "Hall".[4] Their sizes are summarized in Table 1. We compare ATT with other online tensor completion algorithms: TT-FOA [14], TeCPSGD [23], ACP [26], and ATD [26]. TeCPSGD is dependent only on a regularization parameter μ which is set at 0.1. We set the forgetting factor λ at 0.7, 0.7, and 0.5, for ACP, ATD, and TT-FOA, respectively. To have a fair comparison, colour video frames were converted into grayscale ones. The CP-rank, Tucker-rank, and TT-rank were set to 16, $[12, 12, 12]$, and $[6, 6]$, respectively. These tensor rank values were deliberately chosen to ensure that the corresponding TT, CP and Tucker models share the same space (memory) complexity. The experimental results in Tab. 1 and Fig. 8 indicate that the proposed ATT method provided a competitive video completion performance as compared to others. In particular, ATT produced higher

[4]Video sequences: http://jacarini.dinf.usherbrooke.ca/



(a) Full observations

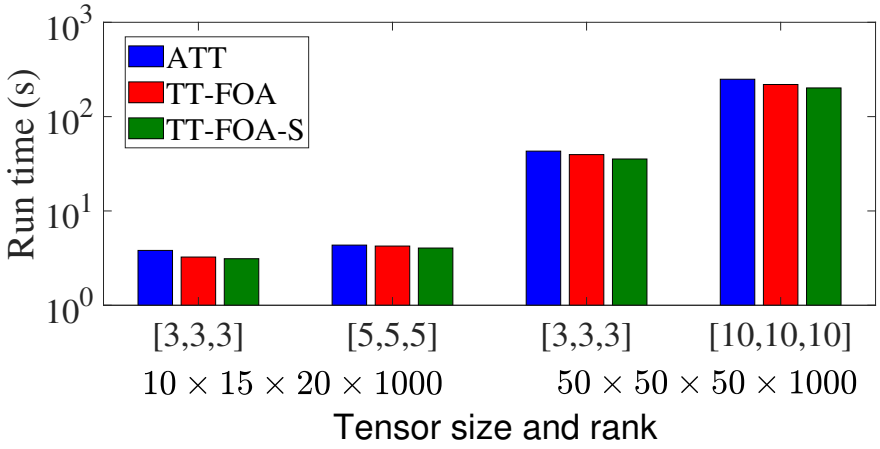
(b) 80% observations

Figure 6: Performance comparison between adaptive tensor-train algorithms: Estimation accuracy.

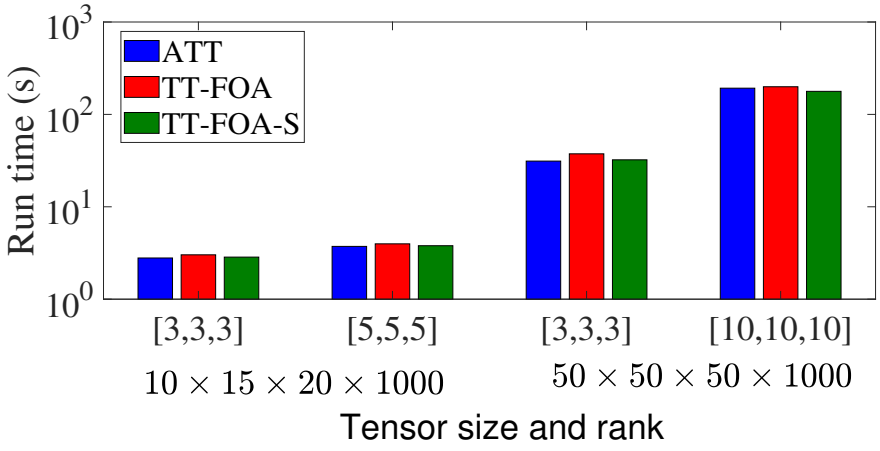
relative errors (RE) than TeCPSGD and ATD, but demonstrated faster runtimes than both methods in most cases. ACP is the fastest adaptive tensor-based method for this application, but its estimation accuracy for cases of highly incomplete observations was lower that of the proposed ATT method, particularly when dealing with Hall and Lobby datasets. TT-FOA was unable to perform online video completion effectively. Notably, with the same space (memory) complexity, the ATT method provides a lower-rank representation for video sequences than the CP and Tucker-based methods. In particular, ATT demonstrated superior efficiency in processing high-dimensional video sequences (e.g., Highway dataset) while offering reasonable estimation accuracy.

5. Conclusions

In this study, we have considered the problem of tensor tracking with missing data under tensor-train format. An effective adaptive tensor-train method called ATT has been proposed. This algorithm is capable of tracking successfully the underlying tensor-train representation of highly incomplete streaming tensors in dynamic environments, even when abrupt changes happen. Its effectiveness for real data has been demonstrated with the online video completion problem. Future works will extend ATT to deal with outliers, impulsive and colored noises.



(a) Full observations



(b) 80% observations

Figure 7: Performance comparison between adaptive tensor-train algorithms: Run time.

Acknowledgments. This work was supported by the National Foundation for Science and Technology Development (NAFOSTED) of Vietnam under Grant 102.04-2021.55.

References

[1] A. Cichocki, N. Lee, I. V. Oseledets, et al., Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions, Found. Trends Mach. Learn. 9 (4-5) (2016) 249–429.

[2] I. V. Oseledets, Tensor-train decomposition, SIAM J. Sci. Comput. 33 (5) (2011) 2295–2317.

[3] S. Holtz, T. Rohwedder, R. Schneider, The alternating linear scheme for tensor optimization in the tensor train format, SIAM J. Sci. Comput. 34 (2) (2012) 683–713.

[4] S. Holtz, T. Rohwedder, R. Schneider, On manifolds of tensors of fixed TT-rank, Numer. Math. 120 (4) (2012) 701–731.

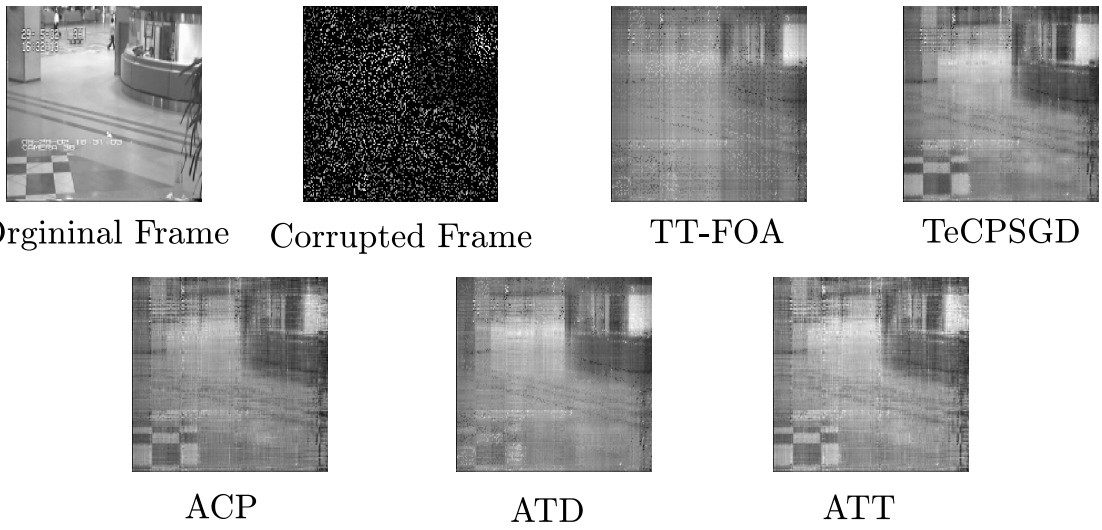


Figure 8: The 500-th video frame of "Hall" data: 80% pixels are missing.

Table 1: Performance of adaptive tensor decompositions on incomplete video sequences.

Dataset	Size	Missing	Online Tensor Completion Methods									
			TT-FOA		TeCPSGD		ACP		ATD		ATT	
			RE	runtime	RE	runtime	RE	runtime	RE	runtime	RE	runtime
Hall	$174 \times 144 \times 3584$	20%	0.2089	38.98(s)	0.1102	40.26(s)	0.1156	10.29(s)	0.1321	53.26(s)	0.1304	39.36(s)
		40%	0.2704	31.58(s)	0.1238	32.64(s)	0.1269	9.89(s)	0.1353	44.25(s)	0.1335	31.12(s)
		80%	0.3159	30.51(s)	0.1362	29.97(s)	0.1494	9.67(s)	0.1404	35.40(s)	0.1417	29.63(s)
Lobby	$128 \times 160 \times 1546$	20%	0.1863	11.02(s)	0.1172	13.75(s)	0.1195	4.09(s)	0.1209	15.40(s)	0.1253	11.42(s)
		40%	0.2174	9.86(s)	0.1288	10.91(s)	0.1363	3.41(s)	0.1228	14.23(s)	0.1312	9.63(s)
		80%	0.2406	8.86(s)	0.1326	9.92(s)	0.1735	2.52(s)	0.1344	11.82(s)	0.1328	8.67(s)
Highway	$320 \times 240 \times 1700$	20%	0.3365	37.39(s)	0.1652	52.43(s)	0.1724	10.40(s)	0.1873	48.20(s)	0.1919	37.63(s)
		40%	0.3913	36.73(s)	0.1851	43.11(s)	0.1902	10.02(s)	0.1986	40.33(s)	0.1975	35.87(s)
		80%	0.4264	28.90(s)	0.1898	34.37(s)	0.1912	9.80(s)	0.1992	32.99(s)	0.2126	26.37(s)

[5] N. Vervliet, O. Debals, L. Sorber, et al., Breaking the Curse of Dimensionality Using Decompositions of Incomplete Tensors: Tensor-based scientific computing in big data analysis, IEEE Signal Process. Mag. 31 (5) (2014) 71–79.

[6] T. Kolajo, O. Daramola, A. Adebiyi, Big data stream analysis: A systematic literature review, J. Big Data 6 (1) (2019) 1–30.
[7] R. J. Little, D. B. Rubin, Statistical Analysis with Missing Data, 2019.
[8] L. T. Thanh, K. Abed-Meraim, N. Linh Trung, A. Hafiane, A contemporary and comprehensive survey on streaming tensor decomposition, IEEE Trans. Knowl. Data Eng. (2022).
[9] C. Lubich, T. Rohwedder, R. Schneider, B. Vandereycken, Dynamical approximation by hierarchical Tucker and tensor-train tensors, SIAM J. Matrix Anal. Appl. 34 (2) (2013) 470–494.
[10] C. Lubich, I. V. Oseledets, B. Vandereycken, Time integration of tensor trains, SIAM J. Numer. Anal. 53 (2) (2015) 917–941.
[11] C. Lubich, B. Vandereycken, H. Walach, Time integration of rank-constrained Tucker tensors, SIAM J. Numer. Anal. 56 (3) (2018) 1273–1290.
[12] H. Liu, L. T. Yang, Y. Guo, X. Xie, J. Ma, An incremental tensor-train decomposition for cyber-physical-social big data, IEEE Trans. Big Data 7 (2) (2021) 341–354.
[13] X. Wang, L. T. Yang, Y. Wang, L. Ren, M. J. Deen, ADTT: A highly efficient distributed tensor-train decomposition method for IIoT big data, IEEE Trans Ind. Inf. 17 (3) (2021) 1573–1582.
[14] L. T. Thanh, K. Abed-Meraim, N. Linh-Trung, R. Boyer, Adaptive algorithms for tracking tensor-train decomposition of streaming tensors, in: Eur. Signal Process. Conf., 2020, pp. 995–999.
[15] Q. Song, H. Ge, J. Caverlee, X. Hu, Tensor completion algorithms in big data analytics, ACM Trans. Knowl. Discov. Data 13 (1) (2019) 1–48.
[16] S. V. Dolgov, B. N. Khoromskij, I. V. Oseledets, D. V. Savostyanov, Computation of extreme eigenvalues in higher dimensions using block tensor train format, Comput. Physics Commun. 185 (4) (2014) 1207–1216.
[17] Q. Zhao, G. Zhou, S. Xie, L. Zhang, A. Cichocki, Tensor ring decomposition, arXiv preprint arXiv:1606.05535 (2016).
[18] Y.-B. Zheng, T.-Z. Huang, X.-L. Zhao, Q. Zhao, T.-X. Jiang, Fully-connected tensor network decomposition and its application to higher-order tensor completion, 2021, pp. 11071–11078.
[19] Z.-C. Wu, T.-Z. Huang, L.-J. Deng, H.-X. Dou, D. Meng, Tensor wheel decomposition and its tensor completion application, in: Adv. Neural Inf. Process. Syst., 2022, pp. 27008–27020.
[20] A. Cichocki, A.-H. Phan, Q. Zhao, et al., Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives, Found. Trends Mach. Learn. 9 (6) (2017) 431–673.
[21] T. Shi, M. Ruth, A. Townsend, Parallel algorithms for computing the tensor-train decomposition, SIAM J. Sci. Comput. 45 (3) (2023) C101–C130.
[22] Y. Zniyed, R. Boyer, A. de Almeida, G. Favier, A TT-Based hierarchical framework for decomposing high-order tensors, SIAM J. Sci. Comput. 42 (2) (2020) 822–848.
[23] M. Mardani, G. Mateos, G. B. Giannakis, Subspace learning and imputation for streaming big data matrices and tensors, IEEE Trans. Signal Process. 63 (10) (2015) 2663–2677.
[24] S. Zhou, N. X. Vinh, J. Bailey, Y. Jia, I. Davidson, Accelerating online CP decompositions for higher order tensors, in: Proc. ACM Int. Conf. Knowl. Discover. Data Min., 2016, pp. 1375–1384.
[25] P. S. Diniz, Adaptive filtering, Springer, 1997.
[26] L. T. Thanh, K. Abed-Meraim, N. L. Trung, A. Hafiane, Tracking online low-rank approximations of higher-order incomplete streaming tensors, Patterns 4 (6) (2023) 100759.