Thèse de Doctorat de

Université d'Orléans

par

Trung Thanh LE

Signal and Image Data Stream Analytics:
From Subspace to Tensor Tracking

Rapporteurs avant soutenance :

Laurent ALBERA	Professeur, Université Rennes 1
Mohammed Nabil EL KORSO	Professeur, Université Paris-Saclay

Composition du Jury :

Président :	Philippe RAVIER	Professeur, Université d'Orléans
Examinateurs :	Laurent ALBERA	Professeur, Université Rennes 1
	Mohammed Nabil EL KORSO	Professeur, Université Paris-Saclay
	Roland BADEAU	Professeur, Télécom ParisTech
	Rémy BOYER	Professeur, Université de Lille
Dir. de thèse :	Karim ABED-MERAIM	Professeur, Université d'Orléans
Co-dir. de thèse :	Adel HAFIANE	MCF-HDR, INSA Centre Val de Loire
Co-encadrant :	Linh Trung NGUYEN	Professeur Associé, Université Nationale due Vietnam - Hanoi

Signal and Image Data Stream Analytics: From Subspace to Tensor Tracking
by Trung Thanh LE

Acknowledgments

This thesis marks the end of my three-year journey towards a Ph.D. degree in Signal Processing from the University of Orléans, INSA CVL, PRISME. Despite having some difficulties and things that didn't go as expected, I have still enjoyed being a doctoral student at the PRISME Laboratory. I am so grateful to have had the opportunity to meet, work with, and learn from so many wonderful people.

To my supervisors. First of all, I would like to express my great appreciation to Prof. Karim Abed-Meraim and Prof. Adel Hafiane, my supervisors at the University of Orléans, INSA CVL, PRISME, and Prof. Nguyen Linh Trung, my advisor at VNU University of Engineering and Technology. I am lucky to work under the supervision of Prof. Karim who is very kind, friendly, and unconditionally supportive throughout my study. Thank you very much for giving me a lot of freedom to pursue my ideas in research. I am so grateful to Prof. Adel for his useful comments on my works in regular meetings. Without the support of Prof. Trung, I wouldn't have got where I am today, thank you so much.

To my colleagues and my friends. Many thanks to my "senpai", Dr. Nguyen Viet Dung, for his suggestions and discussions on my study as well as his academic and social experience in France. Some results in my thesis are specifically inspired from his works on subspace tracking and tensor decomposition. I am very fortunate to have three fantastic friends, Guanglie Ouyang, Zuokun Ouyang, and Pham Minh Tuan, whose sincerity and friendship are highly appreciated. My Ph.D. journey has been much more fun and exciting with their stories, talks, and moments. I would like to thank Dr. Hoang Vy Thuy Lynh for her tremendous support during the years. As starting a new life abroad is never easy, my life in France would not have been as pleasant as it is right now without her help.

To my wife, my parents, and my late little brother. I am immensely thankful to my wife, Vu Thi Kim Chung, for her love, care, and patience. She means everything to me, she is the reason for my happiness and the anchor when things are difficult for me. I am indebted to my parents for all their love and support. Lastly, this thesis is specifically dedicated to the memory of my late little brother, Le Long Xuyen, who has left a void never to be filled in my family.

Thank you!
Le Trung Thanh

Contents

List of Figures

List of Tables

Acronyms

ADMM	Alternating Direction Method of Multipliers
ALS	Alternating Least-Squares
BCD	Block-Coordinate Descent
BTD	Block-term Decomposition
CANDECOMP	Canonical Decompostion
CP	CANDECOMP/PARAFAC
EVD	Eigenvalue Decomposition
FFT, **iFFT**	Fast Fourier transform and its inverse
HDLSS	High Dimension and Low Sample Size
HOOI	Higher Order Orthogonal Iteration
HOSVD	Higher-Order SVD
IoT	Internet of Things
LRA	Low-rank Approximation
PARAFAC	Parallel Factors
PCA	Principal Component Analysis
RE	Relative Error
RLS	Recursive Least Squares
SCM	Sample Covariance Matrix
SEP	Subspace Estimation Performance
ST	Subspace Tracking
SVD	Singular Value Decomposition
TD	Tensor Decomposition
T-SVD	Tensor SVD
TT	Tensor Train

Notations

$\mathbb{R}$ (resp. $\mathbb{C}$)	set of real (resp. complex) numbers
$x, \mathbf{x}, \mathbf{X}, \boldsymbol{\mathcal{X}}$, and $\mathcal{X}$	scalar, vector, matrix, tensor, and set/subset/support
$\mathbf{X}(i,j)/[\mathbf{X}]_{ij}$	(i,j)-th entry of $\mathbf{X}$
$\boldsymbol{\mathcal{X}}(i_1, \ldots, i_N)/[\boldsymbol{\mathcal{X}}]_{i_1 \ldots i_N}$	$(i_1, \ldots, i_N)$-th entry of $\boldsymbol{\mathcal{X}}$
$\mathbf{x} = \mathtt{vec}(\mathbf{X})$	vectorization of $\mathbf{X}$
$\mathbf{X} = \mathtt{diag}(\mathbf{x})$	diagonal matrix $\mathbf{X}$ with $\mathbf{x}$ on the main diagonal
$\mathbf{X}(i,:), \mathbf{X}(:,j)$	i-th row and j-th column of $\mathbf{X}$
$\mathbf{X}^\top, \mathbf{X}^{-1}\, \mathbf{X}^\#$	transpose, inverse, and pseudo-inverse of $\mathbf{X}$
$\lambda_{\max}(\mathbf{X}), \lambda_{\min}(\mathbf{X})$	largest and smallest singular values of $\mathbf{X}$
$\kappa(\mathbf{X})$	condition number of $\mathbf{X}$ equal to $\frac{\lambda_{\max}(\mathbf{X})}{\lambda_{\min}(\mathbf{X})}$
$\mathtt{rank}(\mathbf{X})$	rank of $\mathbf{X}$
$\mathtt{span}(\mathbf{X})$	the column space of a tall matrix $\mathbf{X}$
$\mathtt{tr}(\mathbf{X})$	trace of $\mathbf{X}$
$\theta(\mathbf{X}, \mathbf{Y})$	canonical angle between $\mathtt{span}(\mathbf{X})$ and $\mathtt{span}(\mathbf{Y})$
$\mathbf{I}_n$	$n \times n$ identity matrix
$\mathbf{U}^{(n)}$	n-th loading factor/matrix
$\underline{\mathbf{X}}^{(n)}, \mathtt{unfold}_n(\boldsymbol{\mathcal{X}})$	mode-n unfolding of $\boldsymbol{\mathcal{X}}$
$\boldsymbol{\mathcal{Y}} = \mathtt{bcirc}(\boldsymbol{\mathcal{X}})$	block circulant tensor $\boldsymbol{\mathcal{Y}}$ specified by $\boldsymbol{\mathcal{X}}$
$\circ, \circledast, \odot, \otimes$	outer, Hadamard, Khatri-Rao, and Kronecker product
$\bigodot_{n=1}^{N} \mathbf{U}^{(n)}$	$\mathbf{U}^{(N)} \odot \mathbf{U}^{(N-1)} \odot \cdots \odot \mathbf{U}^{(1)}$
$\bigotimes_{n=1}^{N} \mathbf{U}^{(n)}$	$\mathbf{U}^{(N)} \otimes \mathbf{U}^{(N-1)} \otimes \cdots \otimes \mathbf{U}^{(1)}$
$\boldsymbol{\mathcal{X}} \times_n \mathbf{U}$	n-mode product of $\boldsymbol{\mathcal{X}}$ with $\mathbf{U}$,
$\boldsymbol{\mathcal{X}} \times_n^1 \boldsymbol{\mathcal{Y}}$	mode-$(n,1)$ contracted product of $\boldsymbol{\mathcal{X}}$ with $\boldsymbol{\mathcal{Y}}$
$\boldsymbol{\mathcal{X}} \boxplus_n \boldsymbol{\mathcal{Y}}$	concatenation of $\boldsymbol{\mathcal{X}}$ with $\boldsymbol{\mathcal{Y}}$ along the n-th mode

$\mathcal{X} * \mathcal{Y}$	t-product of $\mathcal{X}$ with $\mathcal{Y}$
$\mathcal{X} \subseteq \mathcal{Y}$	$\mathcal{X}$ is a sub-tensor of $\mathcal{Y}$
$[\![\{\mathbf{U}^{(n)}\}_{n=1}^{N}]\!]$	$\sum_{i=1}^{r} \mathbf{U}^{(1)}(:,i) \circ \mathbf{U}^{(2)}(:,i) \circ \cdots \circ \mathbf{U}^{(N)}(:,i)$
$[\![\mathcal{X}; \{\mathbf{U}^{(n)}\}_{n=1}^{N}]\!]$	$\mathcal{X} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_N \mathbf{U}^{(N)}$
$\|.\|_F, \|.\|_p, \|.\|_*$	Euclidean norm, ℓ_p norm, and nuclear norm
$\lfloor x \rceil$	integer closest to x
$\max\{x, y\}$	maximum of x and y
$\min\{x, y\}$	mminimum of x and y
$(.)_\perp$	orthogonal (perpendicular) complement
$\mathbb{E}[.]$	expectation operator
$\sim$	distributed as
$\propto$	proportional to
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution of mean μ and variance σ^2
$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$	Gaussian vector distribution of mean $\boldsymbol{\mu}$ and variance Σ

General Introduction

<div style="text-align: right; font-size: 3em;">**1**</div>

1.1 Big Data Stream Processing

Stream processing has recently attracted much attention from both academia and industry due to the fact that massive data streams have been increasingly collected over the years and they can be smartly mined to discover new insights and valuable information [1–3]. For example, we are living in the Internet of Things (IoT) era where a huge number of sensing devices have been installed and developed, see Fig. 1.1. These devices have the capability to collect, manage, and transmit data via IoT networks in real time. Accordingly, stream processing is required to retrieve important insights from such IoT data in seconds or even faster for facilitating real-time decision making [4].

In many modern online applications, data streams have three "V" characteristics: *Volume*, *Velocity*, and *Veracity*. As they are continuously generated, their volume grows significantly over time and possibly to infinity. Thus, one of the most notable features of streaming data is that they are unbounded sequences of data samples. *Velocity* refers to the high-speed data arrival rate and real-time processing. Data collected from user interactions in social networks (e.g., Facebook, Instagram, and Twitter) are, for example, at very high velocity. *Veracity* implies the suitability, credibility, and trustworthiness of data streams. More specifically, this characteristic relates to the biasedness, noise, uncertainty, incompleteness, and abnormality in data. Apart from the three "V"s, streaming data have some other distinctive characteristics, including time sensitivity/variation (aka concept drift), heterogeneity (different sources with diversity of data types), volatile and unrepeatable property, and
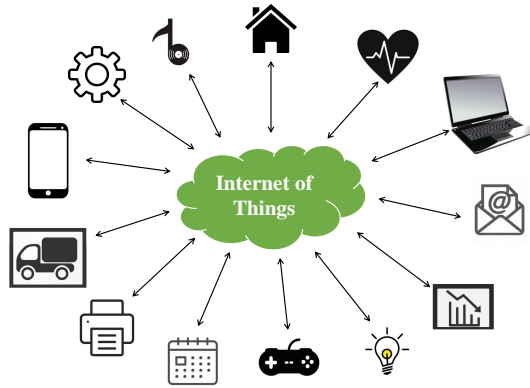
Figure 1.1: Internet of Things

so on [2, 3, 5, 6]. These characteristics lead to several inherent requirements and computational issues for stream processing, such as:

- *Low latency*: Stream methods and systems need to efficiently acquire, manage, and process flows of data without introducing additional delays.

- *Low memory storage*: Stream methods and systems must have the ability to operate in an online fashion with limited memory resources.

- *Scalability*: As streaming data normally grow in size much faster than computational resources, stream processing requires scalable methods and systems.

- *Time variation*: As streaming data can evolve with time, stream methods and systems are required to be capable of tracking their variation along the time.

- *Robustness*: In many cases, streaming data are imperfect and unreliable, so stream methods and systems should have the potential to estimate and compute answers from corrupted observations.

They are, however, also potential benefits of stream processing against batch processing, we refer the readers to Table. 1.1 for a brief comparison between the two kinds of processing.

In this work, we mainly focus on stream methods which are capable of tracking the low-rank approximation (LRA) of big data streams over time. Technically, the primary objective of the LRA is to approximate high-dimensional data by a more compact low-dimensional representation with limited loss of information [7]. Therefore, finding the LRA is a fundamental and essential

Table 1.1: Main differences between batch processing and stream processing

Features	Batch Processing	Stream Processing
Input	Large batches/chunks of data	(Continuous) streams of data
Data size	Known and finite	Unknown and/or infinite
Data type	Static	Dynamic/time-varying
Processing	Process data all at once Process in multiple passes	Process data streams in (near) real time Process in one- or two-pass
Response	Provide after completion	Provide immediately
Hardware	Require much storage Require much processing resources	Require much less storage or no storage Require much less processing resources
Time	Take longer time, latencies in minutes to hours	Take a few seconds or faster

task for data mining in general and streaming data analytics in particular. For the sake of convenience and convention, in what follows, we first list some linear and multilinear algebraic operations (for vectors, matrices, and tensors) that are frequently used throughout this manuscript. Next, we introduce one of the most well-known linear algebra techniques for finding the LRA of matrices in batch setting, singular value decomposition (SVD), and then describe its connection to some common types of tensor decomposition (TD). Finally, we present their online (adaptive) variants for dealing with streaming data derived from one-dimensional observations (i.e., SVD $\rightarrow$ subspace tracking) and multi-dimensional observations (i.e., tensor decomposition $\rightarrow$ tensor tracking).

1.1.1 Vector, Matrix, and Tensor Operations

In this thesis, we use the following notational conventions. Lowercase, boldface lowercase, and boldface capital letters denote scalars (e.g., x), vectors (e.g., $\mathbf{x}$), and matrices (e.g., $\mathbf{X}$), respectively. Calligraphic and bold calligraphic letters are used to represent sets/subsects/supports (e.g., $\mathcal{X}$) and tensors (e.g., $\boldsymbol{\mathcal{X}}$), respectively. For index notations, we use x_i or $\mathbf{x}(i)$ to denote the i-th element of $\mathbf{x}$. The (i, j)-th element, the i-th row, and the j-th column of $\mathbf{X}$ are denoted by $x_{i,j}$ or $\mathbf{X}(i, j)$, $\mathbf{X}_{i,:}$ or $\mathbf{X}(i,:)$, and $\mathbf{X}_{:,j}$ or $\mathbf{X}(:, j)$, respectively. We denote by $\mathbf{X}^{\top}$, $\mathbf{X}^{-1}$, and $\mathbf{X}^{\#}$ the transpose, inverse, and pseudo-inverse of $\mathbf{X}$, respectively. The $(i_1, i_2, \ldots, i_N)$-th element of $\boldsymbol{\mathcal{X}}$ is represented by $x_{i_1, i_2, \ldots, i_N}$, $\boldsymbol{\mathcal{X}}(i_1, i_2, \ldots, i_N)$, or $[\boldsymbol{\mathcal{X}}]_{i_1, i_2, \ldots, i_N}$. In addition, $\boldsymbol{\mathcal{X}}_{:, \ldots, :, i_n, :, \ldots, :}$ or $\boldsymbol{\mathcal{X}}(:, \ldots, :, i_n, :, \ldots, :)$ represents a sub-tensor of $\boldsymbol{\mathcal{X}}$ obtained by holding the n-th index of $\boldsymbol{\mathcal{X}}$ at i_n.

The mode-n matricization of $\mathcal{X}$ is denoted by $\underline{\mathbf{X}}^{(n)}$. Symbols $\|.\|_p$ and $\|.\|_F$ represent the ℓ_p norm and Frobenius norm. In the following, we summarize some useful linear and multilinear algebraic operations, to be used later.

Outer product: Given two vectors $\mathbf{x} \in \mathbb{R}^{N \times 1}$ and $\mathbf{y} \in \mathbb{R}^{M \times 1}$, their outer product is defined as follows

$$
\mathbf{x} \circ \mathbf{y} = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \dots & x_1 y_M \\ x_2 y_1 & x_2 y_2 & \dots & x_2 y_M \\ \vdots & \vdots & \ddots & \vdots \\ x_N y_1 & x_N y_2 & \dots & x_N y_M \end{bmatrix} = \begin{bmatrix} y_1 \mathbf{x} & y_2 \mathbf{x} & \dots & y_M \mathbf{x} \end{bmatrix} \in \mathbb{R}^{N \times M}.
$$

$$(1.1)$$

For a generalized case, the outer product of two tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{J_1 \times J_2 \times \dots \times J_M}$ yields a tensor $\mathcal{Z} = \mathcal{X} \circ \mathcal{Y} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_M}$ with elements

$$
\mathcal{Z}(i_1, i_2, \dots, i_N, j_1, j_2, \dots, j_M) = \mathcal{X}(i_1, i_2, \dots, i_N)\mathcal{Y}(j_1, j_2, \dots, j_M). \quad (1.2)
$$

Kronecker product: Given two matrices $\mathbf{X} \in \mathbb{R}^{N \times M}$ and $\mathbf{Y} \in \mathbb{R}^{P \times Q}$, the Kronecker product of $\mathbf{X}$ and $\mathbf{Y}$ results in an $NP \times MQ$ matrix of the following form

$$
\mathbf{X} \otimes \mathbf{Y} = \begin{bmatrix} x_{1,1}\mathbf{Y} & x_{1,2}\mathbf{Y} & \dots & x_{1,M}\mathbf{Y} \\ x_{2,1}\mathbf{Y} & x_{2,2}\mathbf{Y} & \dots & x_{2,M}\mathbf{Y} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1}\mathbf{Y} & x_{N,2}\mathbf{Y} & \dots & x_{N,M}\mathbf{Y} \end{bmatrix} \in \mathbb{R}^{NP \times MQ}. \quad (1.3)
$$

Khatri-Rao product (aka Column-wise Kronecker product): Given two matrices $\mathbf{Y} \in \mathbb{R}^{N \times r}$ and $\mathbf{Y} \in \mathbb{R}^{M \times r}$, their Khatri-Rao product is an $NM \times r$ matrix of the following form

$$
\mathbf{X} \odot \mathbf{Y} = \begin{bmatrix} \mathbf{X}(:, 1) \otimes \mathbf{Y}(:, 1) & \mathbf{X}(:, 2) \otimes \mathbf{Y}(:, 2) & \dots & \mathbf{X}(:, r) \otimes \mathbf{Y}(:, r) \end{bmatrix} \in \mathbb{R}^{NM \times r}.
$$

$$(1.4)$$

For short, we denote the Kronecker product and Khatri-Rao product of a sequence of matrices $\{\mathbf{U}^{(n)}\}_{n=1}^N$ as follows

$$
\bigotimes_{n=1}^N \mathbf{U}^{(n)} = \mathbf{U}^{(N)} \otimes \mathbf{U}^{(N-1)} \otimes \dots \otimes \mathbf{U}^{(1)}, \quad (1.5)
$$

$$
\bigodot_{n=1}^N \mathbf{U}^{(n)} = \mathbf{U}^{(N)} \odot \mathbf{U}^{(N-1)} \odot \dots \odot \mathbf{U}^{(1)}. \quad (1.6)
$$

Tensor unfold and fold operations: The unfold of $\boldsymbol{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, written as $\mathrm{unfold}(\boldsymbol{X})$, returns a tensor $\boldsymbol{\mathcal{Z}}$ of lower order:

$$\boldsymbol{\mathcal{Z}} = \mathrm{unfold}(\boldsymbol{X}) = \begin{bmatrix} \boldsymbol{X}_{:,\ldots,:,1} \\ \boldsymbol{X}_{:,\ldots,:,2} \\ \vdots \\ \boldsymbol{X}_{:,\ldots,:,I_N} \end{bmatrix} \in \mathbb{R}^{I_1 I_N \times I_2 \times I_3 \times \cdots \times I_{N-1}}. \tag{1.7}$$

Its inverse operator denoted by $\mathrm{fold}(\boldsymbol{\mathcal{Z}})$ reshapes $\boldsymbol{\mathcal{Z}}$ back to $\boldsymbol{X}$ as

$$\mathrm{fold}\big(\mathrm{unfold}(\boldsymbol{X})\big) = \boldsymbol{X}. \tag{1.8}$$

Tensor concatenation: The concatenation of two tensors $\boldsymbol{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times W}$ along the last dimension results in $\boldsymbol{\mathcal{Z}} = \boldsymbol{X} \boxplus \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times (I_N + W)}$ with elements

$$\boldsymbol{\mathcal{Z}}(i_1, i_2, \ldots, i_N) = \begin{cases} \boldsymbol{X}(i_1, i_2, \ldots, i_N), & \text{if } i_N \leq I_N, \\ \boldsymbol{\mathcal{Y}}(i_1, i_2, \ldots, i_N), & \text{if } I_N + W \geq i_N > I_N. \end{cases} \tag{1.9}$$

Mode-n product: The mode-n product of a tensor $\boldsymbol{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ returns a tensor $\boldsymbol{\mathcal{Z}} = \boldsymbol{X} \times_n \mathbf{U} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}$ with elements

$$\boldsymbol{\mathcal{Z}}(i_1, \ldots, i_{n-1}, j, i_{n+1}, \ldots, i_N) = \sum_{i_n=1}^{I_n} \boldsymbol{X}(i_1, \ldots, i_{n-1}, i_n, i_{n+1}, \ldots, i_N) \mathbf{U}(j, i_n). \tag{1.10}$$

The mode-n product of $\boldsymbol{X}$ with N matrices $\{\mathbf{U}^{(n)}\}_{n=1}^{N}$ along all N modes is denoted as

$$\left[\!\left[\boldsymbol{X}, \{\mathbf{U}^{(n)}\}_{n=1}^{N} \right]\!\right] = \boldsymbol{X} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_N \mathbf{U}^{(N)}. \tag{1.11}$$

Mode-$(N, 1)$ product (aka tensor-train contraction): The mode-$(N, 1)$ product of $\boldsymbol{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_N \times J_2 \times \cdots \times J_M}$, written as $\boldsymbol{X} \times_N^1 \boldsymbol{\mathcal{Y}}$, results in a tensor $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{I_1 \times \cdots \times J_{N-1} \times J_2 \times \cdots \times J_M}$ with elements

$$\boldsymbol{\mathcal{Z}}(i_1, \ldots, i_{N-1}, j_2, \ldots, j_M) = \sum_{i_N=1}^{I_N} \boldsymbol{X}(i_1, \ldots, i_{N-1}, i_N) \boldsymbol{\mathcal{Y}}(i_N, j_2, \ldots, j_M). \tag{1.12}$$

T-product: The t-product of $\boldsymbol{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_2 \times J \times I_3 \times \cdots \times I_N}$, written as $\boldsymbol{X} * \boldsymbol{\mathcal{Y}}$, returns an $I_1 \times J \times I_3 \times \cdots \times I_N$ tensor $\boldsymbol{\mathcal{Z}}$ of the recursive form

$$\boldsymbol{\mathcal{Z}} = \boldsymbol{X} * \boldsymbol{\mathcal{Y}} = \mathrm{fold}\big(\mathrm{bcirc}(\boldsymbol{X}) * \mathrm{unfold}(\boldsymbol{\mathcal{Y}})\big), \tag{1.13}$$

where $\texttt{circ}(.)$ is a block circulant tensor defined as

$$
\texttt{bcirc}(\boldsymbol{\mathcal{U}}) =
\begin{bmatrix}
\boldsymbol{\mathcal{U}}_1 & \boldsymbol{\mathcal{U}}_{I_N} & \boldsymbol{\mathcal{U}}_{I_{N-1}} & \dots & \boldsymbol{\mathcal{U}}_2 \\
\boldsymbol{\mathcal{U}}_2 & \boldsymbol{\mathcal{U}}_1 & \boldsymbol{\mathcal{U}}_{I_N} & \dots & \boldsymbol{\mathcal{U}}_3 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
\boldsymbol{\mathcal{U}}_{I_N} & \boldsymbol{\mathcal{U}}_{I_{N-1}} & \dots & \boldsymbol{\mathcal{U}}_2 & \boldsymbol{\mathcal{U}}_1
\end{bmatrix}, \tag{1.14}
$$

where $\boldsymbol{\mathcal{U}}_i = \boldsymbol{\mathcal{U}}_{:,\dots,:,i}$ and the base case of the t-product of two 3-order tensors $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_2 \times K \times J_3}$ is defined as

$$
\boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}} = \texttt{fold}\big(\texttt{bcirc}(\boldsymbol{\mathcal{A}}) \cdot \texttt{unfold}(\boldsymbol{\mathcal{B}})\big) \in \mathbb{R}^{J_1 \times K \times J_3}. \tag{1.15}
$$

Inner product: Given two tensors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ of the same size $I_1 \times I_2 \times \cdots \times I_N$, their inner product is defined as

$$
\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i_N}^{I_N} \boldsymbol{\mathcal{X}}(i_1, i_2 \dots, i_N) \boldsymbol{\mathcal{Y}}(i_1, i_2 \dots, i_N). \tag{1.16}
$$

1.1.2 Batch Low-rank Approximation: From SVD to Tensor Decomposition

It is very well known that SVD is one of the most powerful and widely-used linear algebra techniques with a number of applications in various domains [8, 9]. Particularly, the compact SVD of a rank-r matrix $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$ is given by

$$
\mathbf{X} \overset{\text{SVD}}{=} \underbrace{\big[\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_r\big]}_{\mathbf{U}} \underbrace{\begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_r \end{bmatrix}}_{\boldsymbol{\Lambda}} \underbrace{\begin{bmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \vdots \\ \mathbf{v}_r^\top \end{bmatrix}}_{\mathbf{V}^\top} = \sum_{i=1}^{r} \lambda_i \mathbf{u}_i \mathbf{v}_i^\top, \tag{1.17}
$$

where $\mathbf{U} \in \mathbb{R}^{I_1 \times r}$ and $\mathbf{V} \in \mathbb{R}^{I_2 \times r}$ are unitary matrices; and $\boldsymbol{\Lambda} \in \mathbb{R}^{r \times r}$ is a diagonal matrix whose diagonal values are positive, i.e., $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > 0$, see Fig. 1.2 for an illustration. For the problem of low-rank approximation in batch setting, the following theorem indicates that SVD can give the best LRA for any matrix $\mathbf{X}$.

Figure 1.2: SVD of a rank-r matrix X.

Theorem 1 (Eckart-Young-Mirsky Theorem [9]) *Denote by* $\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^\top$ *the SVD of* $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$. *If* $k \leq \mathrm{rank}(\mathbf{X})$ *and* $\mathbf{X}_k = \sum_{i=1}^{k} \lambda_i \mathbf{u}_i \mathbf{v}_i^\top$, *then*

$$\min_{\substack{\mathbf{A} \in \mathbb{R}^{I_1 \times I_2} \\ \mathrm{rank}(\mathbf{A}) \leq k}} \left\| \mathbf{X} - \mathbf{A} \right\| = \left\| \mathbf{X} - \mathbf{X}_k \right\|, \tag{1.18}$$

with respect to both the spectral norm and Frobenius norm.

Thanks to Theorem 1, the best rank-k approximation of $\mathbf{X}$ can be obtained by applying the following procedure:

- **Step 1**: Compute $\mathbf{X} \overset{\text{SVD}}{=} \mathbf{U}\mathbf{\Lambda}\mathbf{V}^\top$, where $\mathbf{U} \in \mathbb{R}^{I_1 \times I_1}$ and $\mathbf{V} \in \mathbb{R}^{I_2 \times I_2}$ are unitary matrices, and the diagonal matrix $\mathbf{\Lambda} \in \mathbb{R}^{I_1 \times I_2}$ contains positive diagonal entries in descending order.

- **Step 2**: Select the first k singular vectors from $\mathbf{U}$ and $\mathbf{V}$ to form the following matrices $\mathbf{U}_k = \mathbf{U}(:, 1:k)$ and $\mathbf{V}_k = \mathbf{V}(:, 1:k)$.

- **Step 3**: Select the top k strongest singular values in $\mathbf{\Lambda}$ to form: $\mathbf{\Lambda}_k = \mathbf{\Lambda}(1:k, 1:k)$.

- **Step 4**: Derive the best rank-k approximation of $\mathbf{X}$ from: $\mathbf{X}_k = \mathbf{U}_k \mathbf{\Lambda}_k \mathbf{V}_k^\top$.

When dealing with tensors (aka, multidimensional arrays), several multiway extensions of the SVD have been developed for tensor decomposition (TD) in the literature [10–13]. The five common types of TD are CP/PARFAC [14], Tucker/HOSVD [15], tensor-train/network [16], t-SVD [17], and block-term decomposition (BTD) [18], see Fig. 1.3 for illustrations. Specifically, they aim to factorize a tensor into a set of basis components (e.g., vectors, matrices, or simpler tensors) and hence offer good low-rank tensor approximations. In the following, we describe their connection to SVD and refer the readers to Chapter 5 for further details on their main features, properties, and algorithms.

CP/PARAFAC Decomposition: Similar to SVD that represents $\mathbf{X}$ by a sum of rank-1 matrices (i.e., $\lambda_i \mathbf{u}_i \mathbf{v}_i^\top$), the CP decomposition also factorizes a tensor
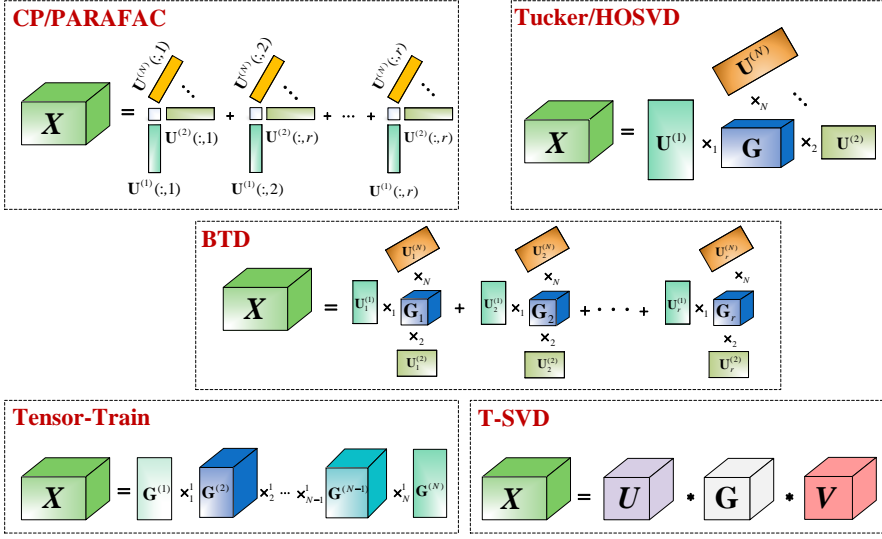
Figure 1.3: Multiway extensions of SVD to high-order tensors: CP/PARAFAC, Tucker, BTD, tensor-train, and t-SVD.

$\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ into rank-1 terms:

$$\mathcal{X} \overset{\text{CP}}{=} \sum_{i=1}^{r} \lambda_i \underbrace{\mathbf{u}_i^{(1)} \circ \mathbf{u}_i^{(2)} \circ \cdots \circ \mathbf{u}_i^{(N)}}_{\text{rank-1 term}}, \tag{1.19}$$

where $\mathbf{u}_i^{(n)} \in \mathbb{R}^{I_n \times 1}$ with $1 \leq n \leq N$ plays the same role as singular vectors of $\mathbf{U}$ and $\mathbf{V}$ in the SVD model (1.17) (note that $\mathbf{u}_i \mathbf{v}_i^\top = \mathbf{u}_i \circ \mathbf{v}_i$) [14]. The matrix $\mathbf{U}^{(n)} = \left[\mathbf{u}_1^{(n)}, \mathbf{u}_2^{(n)}, \ldots, \mathbf{u}_r^{(n)} \right]$ is the n-th CP factor of $\mathcal{X}$ and it is not required to be orthogonal. Following the general definition of matrix rank, the smallest integer r satisfying (1.19) is referred to as the tensor (CP) rank of $\mathcal{X}$. Under certain conditions, CP decomposition is essentially unique up to a permutation and scale which is an useful property in many applications.

Tucker/HOSVD Decomposition: Apart from the classical form (1.17), we can express the SVD of $\mathbf{X}$ as follows

$$\mathbf{X} \overset{\text{SVD}}{=} \mathbf{U}\mathbf{\Lambda}\mathbf{V}^\top = \underbrace{\mathbf{\Lambda}}_{\text{core}} \underbrace{\times_1 \mathbf{U} \times_2 \mathbf{V}}_{\text{2 factors}}. \tag{1.20}$$

Accordingly, a direct multiway extension of (1.20) to high-order tensors can be given by

$$\mathcal{X} \overset{\text{Tucker}}{=} \underbrace{\mathcal{G}}_{\text{core}} \underbrace{\times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_N \mathbf{U}^{(N)}}_{N \text{ factors}}, \tag{1.21}$$

where the core $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_N}$ is a tensor of smaller size than $\mathcal{X}$ (i.e., $r_n \leq I_n \ \forall n$) and N tensor factors $\{\mathbf{U}^{(n)}\}_{n=1}^{N}$, $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times r_n}$ are orthogonal matrices. The representation model (1.21) is regarded as the high-order SVD (HOSVD) or Tucker format [15]. Unlike the SVD and CP, Tucker/HOSVD is not unique in general. However, as the subspace covering $\mathbf{U}^{(n)}$ is physically unique, its main objective is for finding principal subspaces of the tensor factors [11].

Block-Term Decomposition: BTD factorizes $\mathcal{X}$ into several blocks of low multilinear-rank instead of rank-1 terms

$$\mathcal{X} \stackrel{\text{BTD}}{=} \sum_{i=1}^{r} \underbrace{\mathcal{G}_i \times_1 \mathbf{U}_i^{(1)} \times_2 \mathbf{U}_i^{(2)} \times_3 \cdots \times_N \mathbf{U}_i^{(N)}}_{\text{low multilinear-rank term}}. \qquad (1.22)$$

The BTD can be viewed as a unification and generalization of the two well-known CP and Tucker decompositions. Specifically, when $\{\mathcal{G}_i\}_{i=1}^{r}$ are diagonal tensors, BTD boils down to the CP decomposition. It has the form of Tucker decomposition when only one block term (i.e., $r = 1$) is considered. In addition, several appealing features of the BTD are inherited from CP and Tucker such as stable computation of Tucker, identification and uniqueness of CP [18]. In parallel, it is worth recalling a remark in [18] that "the rank of a higher-order tensor is actually a combination of the two aspects: one should specify the number of blocks and their size". That means BTD provides a unified approach to generalize the concept of matrix rank to tensors.

Tensor-Train Decomposition: Together with (1.17) and (1.20), we can write the SVD of $\mathbf{X}$ as

$$\mathbf{X}(i_1, i_2) \stackrel{\text{SVD}}{=} \sum_{k=1}^{r} \lambda_k \mathbf{U}(i_1, k) \mathbf{V}(k, i_2). \qquad (1.23)$$

Accordingly, each element of a high-order tensor $\mathcal{X}$ can be represented by

$$\mathcal{X}(i_1, i_2, \dots, i_N) \stackrel{\text{TT}}{=} \sum_{k_1, k_2, \dots, k_{N-1}}^{r_1, r_2, \dots, r_{N-1}} \mathcal{G}_1(1, i_1, k_1) \mathcal{G}_2(k_1, i_2, k_2) \dots \mathcal{G}_N(k_{N-1}, i_N, 1).$$

$$(1.24)$$

where $\mathcal{G}_n$ is an $r_{n-1} \times I_n \times r_n$ tensor with $n = 1, 2, \dots, N-1$ and $r_0 = r_N = 1$. We refer to the representation model (1.24) as tensor-train (TT). Like CP, the TT format offers a memory-saving model for representing high-order tensors. Like Tucker, the TT decomposition and the TT rank $\mathbf{r} = [r_1, r_2, \dots, r_{N-1}]$ of any tensor $\mathcal{X}$ can be numerically computed in a stable and efficient way.

t-SVD Decomposition: Last but not least, another extension of SVD to high-order tensors is the tensor SVD (t-SVD) which is of the following form:

$$\mathcal{X} \stackrel{\text{t-SVD}}{=} \underbrace{\mathcal{U}}_{\text{orthogonal}} * \underbrace{\mathcal{G}}_{f\text{-diagonal}} * \underbrace{\mathcal{V}}_{\text{orthogonal}}, \qquad (1.25)$$

where $\mathcal{U}$ and $\mathcal{V}$ are unitary tensors, and $\mathcal{G}$ is a rectangle f-diagonal tensor whose frontal slices are diagonal matrices [17]. Intuitively, the t-SVD model (1.25) shares the similar form with the SVD in (1.17). However, due to the t-product "$*$", the algebraic framework used in the t-SVD is quite different from the classical (multi)-linear algebra in other types of TD and SVD. For example, most of its computations are performed in the Fourier domain. Under the t-SVD format, the tubal-rank which is equal to the number of non-zero tubes of $\mathcal{G}$ is used to define the LRA of tensors in the same manner as the SVD.

1.1.3   Online Low-rank Approximation: From Subspace to Tensor Tracking

In online setting, data samples are continuously collected with time. Accordingly, recomputing the batch LRA methods (e.g., SVD or batch TD algorithms) at each time step becomes inefficient due to their high complexity and the time variation, aka concept/distribution drift. This has led to defining a variant of the LRA called online (adaptive) LRA in which we may want to track the underlying process that generates streaming data with time.

When observations arriving at each time are one-dimensional (i.e., vectors), the main interest in the online LRA is to estimate the principal subspace that compactly spans these observations over time. Specifically, it is referred to as the problem of subspace tracking (ST) in signal processing which has been developed for over three decades [19–21]. In general, on the arrival of the new data $\mathbf{y}_t \in \mathbb{R}^{I_1 \times 1}$ at time t, the subspace matrix $\mathbf{U}_t \in \mathbb{R}^{I_1 \times r}$ can be derived from analysing the spectrum of the following covariance matrix

$$\mathbf{C}_t = \sum_{\tau=t-L_t+1}^{t} \beta^{t-\tau} \mathbf{y}_\tau \mathbf{y}_\tau^\top, \tag{1.26}$$

where L_t is the window length and $0 < \beta \le 1$ is the forgetting factor [20]. When $L_t = t$ and $\beta = 1$, $\mathbf{C}_t$ in (1.26) boils down to the classical sample covariance matrix. More specifically, in a connection to the batch LRA using SVD, the vector $\mathbf{y}_t$ can be seen as the t-th column of the underlying matrix $\mathbf{X}_t = [\mathbf{X}_{t-1} \ \mathbf{y}_t]$, see Fig. 1.4 for an illustration. The subspace matrix $\mathbf{U}_t$ plays a role as the left singular vector matrix of $\mathbf{X}_t$, while the coefficient vector $\mathbf{w}_t = \mathbf{U}_t^\top \mathbf{y}_t$ is indeed the t-th row of the matrix $\mathbf{V}\mathbf{\Lambda}$ in the SVD expression (1.17). Depending on the choice of $\mathbf{C}_t$ and the subspace estimation technique, we can obtain several subspace tracking algorithms.

When observations arriving at each time are multidimensional (i.e., tensors), the online LRA turns out to be tensor tracking which can be considered as a generalization of subspace tracking. In particular, we wish to estimate
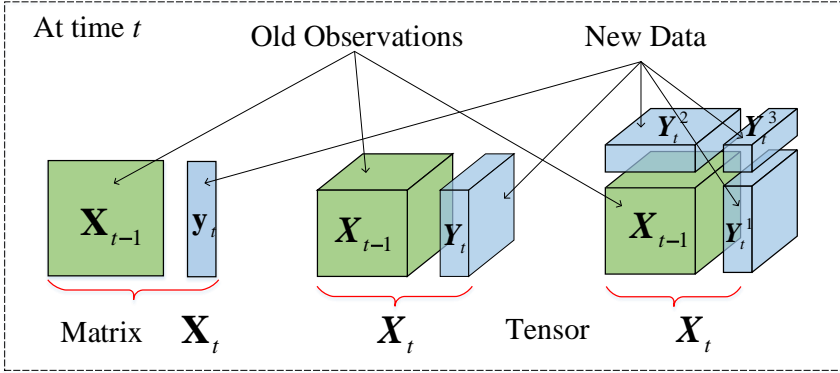
Figure 1.4: Streaming data.

the tensor dictionary (e.g., core tensor(s) and tensor factors) that generates the underlying streaming data $\mathcal{X}_t$ over time:

$$\mathcal{X}_t = \begin{cases} \mathcal{X}_{t-1} \boxplus \mathcal{Y}_t & \text{if single-aspect streaming} \\ \mathcal{X}_{t-1} \cup \mathcal{Y}_t & \text{if multi-aspect streaming} \end{cases}, \qquad (1.27)$$

where "$\boxplus$" and "$\cup$" denote the tensor concatenation and union operator, while $\mathcal{X}_{t-1}$ and $\mathcal{Y}_t$ represent the old and new observations, respectively. The "single-aspect streaming" model and the "multi-aspect streaming" model are, respectively, dedicated to represent data streams having one dimension and multiple dimensions varying with time. When new data samples arrive, the tensor dictionary of $\mathcal{X}_t$ should be incrementally updated without reusing the batch TD algorithms. Similar to subspace tracking, we can also obtain many tensor tracking algorithms based on different tensor formats, streaming models, and optimization techniques. The readers are referred to Chapter 5 for a comprehensive overview of the state-of-the-art tensor tracking algorithms.

In recent years, the explosion of big data streams have posed significant challenges to the online LRA problem. For example, efficiency and robustness are highly important when we deal with streaming data in high dimensions. Many theoretical results in random matrix theory (e.g., [22–24]) indicated that the sample covariance matrix (SCM) is not an efficient estimator of the actual covariance matrix in the high-dimension, low-sample-size regime where datasets are massive in both dimension and sample size. However, most of the state-of-the-art subspace tracking methods in the literature are mainly based on the spectral analysis of the SCM, and thus, they are not effective in such a regime. In parallel, sparse outliers and missing data become more and more ubiquitous in modern streaming applications [6]. Sparse outliers
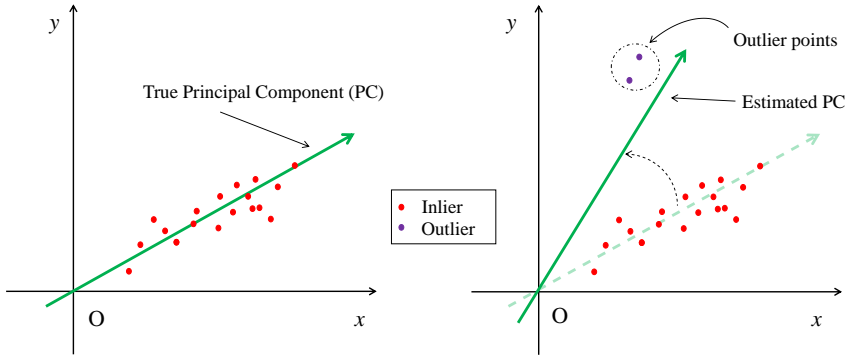
Figure 1.5: Effect of outliers on the standard PCA.

are data points that appear to be inconsistent with or exhibit abnormal behaviour different from others. Missing data are often encountered during the acquisition and collection. Both sparse outliers and missing data can cause several issues for knowledge discovery from data in general and data streams in particular, see Fig. 1.5 for an illustration of outlier's impact on the standard principal component analysis (PCA) which specifically uses SVD in its computation. Therefore, it requires robust algorithms capable of dealing with such data corruptions with time. In addition, scalable tracking algorithms are always desirable for handling modern data streams, especially dealing with large-scale and high-multidimensional data streams. As indicated later, most of the existing tracking algorithms are of high complexity with respect to both computation and memory storage. Accordingly, it is essential to develop efficient and scalable tracking techniques of low cost. In this work, we aim to develop efficient and effective tracking algorithms which have the capability to deal with such challenges.

1.2   Thesis Description

1.2.1   Thesis Outline and Contributions

The rest of my thesis is organized into two major parts addressing respectively subspace tracking and tensor tracking, followed by the conclusion and outlook, please see Fig. 1.6 for an overview.
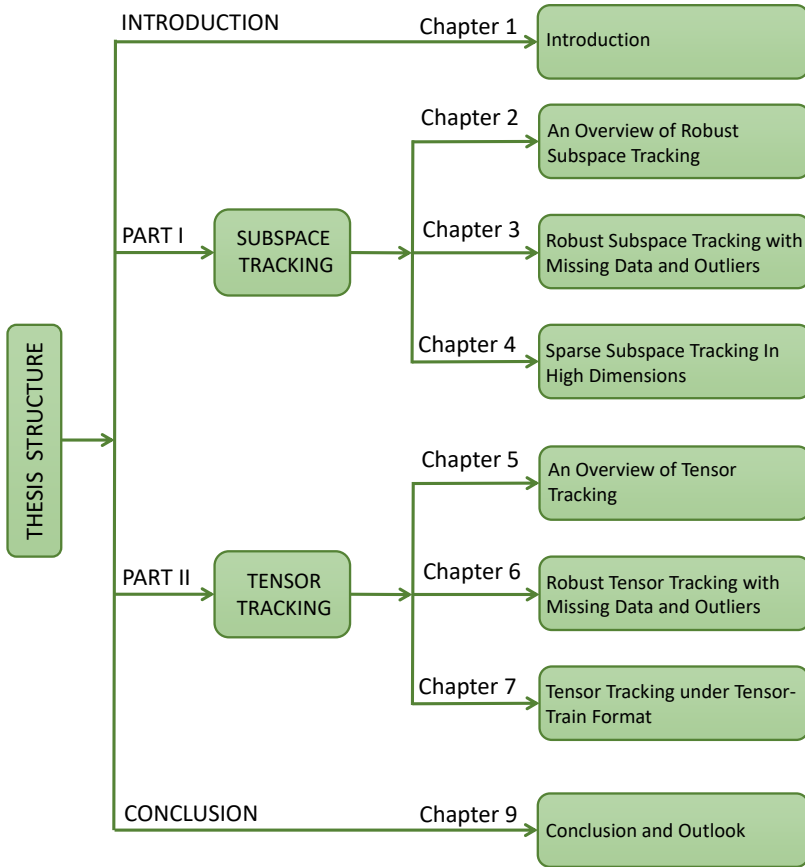
Figure 1.6: Thesis structure.

Part I: Subspace Tracking

In Chapter 2, we provide a brief survey on recent robust subspace tracking algorithms which were mostly developed over the last decade. Particularly, we begin by introducing the basic ideas of the subspace tracking problem. We then highlight main classes of algorithms for dealing with non-Gaussian noises (e.g., sparse outliers, impulsive noise, and colored noise). Recent years have also witnessed the widespread of high-dimensional data analysis in which sparse representation-based methods are successfully applied to many signal processing applications. Accordingly, the state-of-the-art sparse subspace tracking algorithms are also reviewed therein.

In Chapter 3, we propose a novel algorithm, namely PETRELS-ADMM, to deal with subspace tracking in the presence of outliers and missing data. The proposed approach consists of two main stages: outlier rejection and sub-

space estimation. In the first stage, alternating direction method of multipliers (ADMM) is effectively exploited to detect outliers affecting the observed data. In the second stage, we propose an improved version of the parallel estimation and tracking by recursive least squares (PETRELS) algorithm to update the underlying subspace in the missing data context. We then present a theoretical convergence analysis of PETRELS-ADMM which shows that it generates a sequence of subspace solutions converging to the optimum of its batch counterpart. The effectiveness of the proposed algorithm, as compared to state-of-the-art algorithms, is illustrated on both simulated and real data.

In Chapter 4, we develop a new provable effective method called OPIT for tracking the sparse principal subspace of data streams over time. Particularly, OPIT introduces a new adaptive variant of power iteration with space and computational complexity linear to the data dimension. In addition, a new column-based thresholding operator is developed to regularize the subspace sparsity. Utilizing both advantages of power iteration and thresholding operation, OPIT is capable of tracking the underlying subspace in both classical regime and high dimensional regime. We also present a theoretical result on its convergence to verify its consistency in high dimensions. Several experiments are carried out on both synthetic and real data to demonstrate the tracking ability of OPIT.

Part II: Tensor Tracking

In Chapter 5, we provide a contemporary and comprehensive survey on different types of tensor tracking techniques. We particularly categorize the state-of-the-art methods into three main groups: streaming CP decompositions, streaming Tucker decompositions, and streaming decompositions under other tensor formats (i.e., tensor-train, t-SVD, and BTD). In each group, we further divide the existing algorithms into sub-categories based on their main optimization framework and model architectures. Specifically, 04 main groups of streaming CP decomposition algorithms were emphasized, including subspace based, block-coordinate descent, Bayesian inference, and multi-aspect streaming decompositions. We categorized streaming Tucker decomposition algorithms into three major classes based on their model architecture. They are online tensor dictionary learning, tensor subspace tracking, and multi-aspect streaming decompositions. Finally, a brief survey on the existing methods which are capable of tracking tensors under TT, BTD, and t-SVD formats is presented.

In Chapter 6, we propose three novel adaptive algorithms for tracking higher-order streaming tensors with time, including ACP, ATD, and RACP. Under the CP format, ACP minimizes an exponentially weighted recursive least-squares cost function to obtain the tensor factors in an efficient way,

thanks to the alternative minimization framework and the randomized sketching technique. Under the Tucker format, ATD first tracks the underlying low-dimensional subspaces covering the tensor factors, and then estimates the core tensor using a stochastic approximation. Both the two algorithms ACP and ATD are fast and fully capable of tracking streaming tensors from incomplete observations. When observations are corrupted by sparse outliers, we introduce the so-called RACP algorithm robust to gross corruptions. Particularly, RACP first performs online outlier rejection to accurately detect and remove sparse outliers, and then performs tensor factor tracking to efficiently update the tensor factors. Convergence analysis for three algorithms are established in the sense that the sequence of generated solutions converges asymptotically to a stationary point of the objective function. Extensive experiments are conducted on both synthetic and real data to demonstrate the effectiveness of the proposed algorithms in comparison with state-of-the-art adaptive algorithms.

In Chapter 7, we introduce three new methods for the problem of streaming tensor-train decomposition. The first method called TT-FOA is capable of tracking the low-rank components of high-order tensors from noisy and high-dimensional data with high accuracy, even when they come from time-dependent observations. The second method called ATT is particularly designed for handling incomplete streaming tensors. ATT is scalable, effective, and adept at estimating low TT-rank component of streaming tensors. Besides, ATT can support parallel and distributed computing. To deal with sparse outliers, we propose the so-called ROBOT which stands for ROBust Online Tensor-Train decomposition. Technically, ROBOT has the ability to tracking streaming tensors from imperfect streams (i.e., due to noise, outliers, and missing data) as well as tracking their time variation in dynamic environments.

Conclusion and Outlook

Chapter 8 concludes the thesis with our main results and an outlook to future works. Particularly, we present several research challenges and open problems that should be considered for the development of tracking the low-rank component of data streams in the future. They are data imperfection and corruption; rank revealing and tracking; efficient and scalable tensor tracking; and other aspects such as theoretical analysis, symbolic data, and tracking under some less common tensor formats. Possible solutions for these challenges are also discussed.

1.2.2 List of Publications

Most of the above results have been published/submitted in the following papers:

Journal Papers:

[25] L. T. Thanh, N. V. Dung, N. L. Trung and K. Abed-Meraim, *"Robust Subspace Tracking With Missing Data and Outliers: Novel Algorithm With Convergence Guarantee"*, **IEEE Trans. Signal Process.**, vol. 69, pp. 2070–2085, 2021.

[26] L. T. Thanh, N. V. Dung, N. L. Trung and K. Abed-Meraim, *"Robust Subspace Tracking Algorithms in Signal Processing: A Brief Survey"*, **REV J. Elect. Commun.**, vol. 11, no. 1–2, pp. 15–25, 2021.

[27] L. T. Thanh, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"Robust Tensor Tracking with Missing Data and Outliers: Novel Adaptive CP Decomposition and Convergence Analysis"*, **IEEE Trans. Signal Process.**, vol. 70, pp. 4305–4320, 2022.

[28] L. T. Thanh, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"A Contemporary and Comprehensive Survey on Streaming Tensor Decomposition"*, **IEEE Trans. Knowl. Data. Eng.**, 2022 (in press).

[29] L. T. Thanh, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"OPIT: A Simple and Effective Method for Sparse Subspace Tracking in High-dimension and Low-sample-size Context"*, **IEEE Trans. Signal Process.**, 2022 (submitted).

[30] L. T. Thanh, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"Tracking Online Low-Rank Approximations of Higher-Order Incomplete Streaming Tensors"*, **Elsevier Patterns**, 2022 (submitted).

[31] L. T. Thanh, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"Streaming Tensor-Train Decomposition With Missing Data"*, **Elsevier Signal Process.**, 2022 (submitted).

Conference Papers:

[32] L. T. Thanh, K. Abed-Meraim, N. L. Trung and R. Boyer, *"Adaptive Algorithms for Tracking Tensor-Train Decomposition of Streaming Tensors"*, in **Proc. 28th EUSIPCO**, 2020, pp. 995–999.

[33] L. T. Thanh, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"A Fast Randomized Adaptive CP Decomposition for Streaming Tensors"*, in **Proc. 46th ICASSP**, 2021, pp. 2910–2914.

[34] L. T. Thanh, K. Abed-Meraim, A. Hafiane and N. L. Trung, *"Sparse Subspace Tracking in High Dimensions"*, in **Proc. 47th ICASSP**, 2022, pp. 5892–5896.

[35] L. T. Thanh, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"Robust Tensor Tracking With Missing Data Under Tensor-Train Format"*, in **Proc. 30th EUSIPCO**, 2022, pp. 832–836.

[36] L. T. Thanh, T. T. Duy, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"Robust Online Tucker Dictionary Learning from Multidimensional Data Streams"*, in **Proc. 14th APSIPA-ASC**, 2022, pp. 1815–1820.

Contributions Outside the Scope of the Thesis

During my Ph.D study, I have also some other contributions to system identification which are not included in this thesis:

[37] <u>L. T. Thanh</u>, K. Abed-Meraim and N. L. Trung, *"Misspecified Cramer-Rao Bounds for Blind Channel Estimation under Channel Order Misspecification"*, **IEEE Trans. Signal Process.**, vol. 69, pp. 5372–5385, 2021.

[38] <u>L. T. Thanh</u>, K. Abed-Meraim and N. L. Trung, *"Performance Lower Bounds of Blind System Identification Techniques in the Presence of Channel Order Estimation Error"*, in **Proc. 29th EUSIPCO**, 2021, pp. 1646–1650.

[39] O. Rekik, A. Mokraoui, T. T. T Quynh, <u>L. T. Thanh</u> and K. Abed-Meraim. *"Side Information Effect on Semi-Blind Channel Identification for MIMO-OFDM Communications Systems"*, in **Proc. 55th ASILOMAR 2021**, pp. 443–448.

Particularly in [37, 38], we have addressed the problem of analyzing the theoretical performance limit of system identification techniques under the misspecification of the channel order through the lens of the misspecified Cramer-Rao bound (MCRB) – which is an extension of the well-known Cramer-Rao bound (CRB) when the underlying system model is misspecified. Specifically, we have introduced a new interpretation of the MCRB, called the generalized MCRB (GMCRB), via the Moore–Penrose inverse operator. This bound is useful for singular problems and particularly blind channel estimation problems in which the Hessian matrix is noninvertible. Two closed-form expressions of the GMCRB are derived for unbiased blind estimators when the channel order is misspecified. The first bound deals with deterministic models where both the channel and unknown symbols are deterministic. The second one is devoted to stochastic models where we assume that transmitted symbols are unknown random variables i.i.d. drawn from a Gaussian distribution. Two case studies of channel order misspecification are investigated to demonstrate the effectiveness of the proposed GMCRBs over the classical CRBs. In [39], we have investigated the effect of different prior about communications channels (e.g., specular channel model, finite memory linear time invariant channel model, misspecification caused by array calibration errors, so on) on the performance of semi-blind channel identification for MIMO-OFDM systems.

An Overview of Robust Subspace Tracking

2

Principal component analysis (PCA) and subspace estimation (SE) are popular data analysis tools and used in a wide range of applications. The main interest in PCA/SE is for dimensionality reduction and low-rank approximation purposes. The emergence of big data streams have led to several essential issues for performing PCA/SE. Among them are (i) the size of such data streams increases over time, (ii) the underlying models may be time-dependent, and (iii) problem of dealing with the uncertainty and incompleteness in data. A robust variant of PCA/SE for such data streams, namely robust online PCA or robust subspace tracking (RST), has been introduced as a good alternative. The main

goal of this chapter is to provide a brief survey on recent RST algorithms in signal processing. Particularly, we begin this survey by introducing the basic ideas of the RST problem. Then, different aspects of RST are reviewed with respect to different kinds of non-Gaussian noises and sparse constraints. Our own contributions on this topic are also highlighted.

2.1 Introduction

Principal component analysis (PCA) and subspace estimation (SE) are widely used as a fundamental step for dimensionality reduction and analysis. Their main purpose is to extract low-dimensional subspaces from high-dimensional data while still keeping as much relevant information as possible. Consequently, PCA and SE have found success in a wide range of fields, from finance to neuroscience, with the most successful applications in computer science. The main difference between them is that PCA emphasizes the use of eigenvectors rather than of subspace as in SE. PCA in a standard set-up can be implemented by using either eigenvalue decomposition (EVD) or singular value decomposition (SVD) and is proved to be optimal in terms of the Frobenius-norm approximation error by the Eckart-Young theorem [40].

Recent years have witnessed an increasing interest in adaptive processing [2]. It is mainly due to the fact that online applications generate a huge amount of data streams over time and such streams are often with high veracity and velocity. It is known that veracity requires robust algorithms for handling imperfect data while velocity demands (near) real-time processing. Accordingly, important classes of PCA, such as subspace tracking (ST) also called PCA for streaming data or streaming PCA or dynamic PCA, and ST with missing data have drawn much research attention recently in signal processing and modern data analysis.

The attractive point of ST resides on two aspects. First, in a similar manner to batch subspace methods [20], both the main components and the disturbance components of data observation can be exploited in many different ways. In fact, the subspace is simple to understand (i.e., in a statistical sense) and implement, thus proving its efficiency in many practical applications. Second, different from batch subspace methods, ST has a better trade-off between the accuracy and the computational complexity, thus making it suitable for time-sensitivity and real-time applications. Due to its practical use, we can find a wide range of applications in diverse fields [19, 20, 41], for example, direction of arrival (DoA) tracking in radar and sonar, data compression and filtering, blind channel estimation and equalization, and pattern recognition, to name a few.

However, it is well-known that PCA/SE is very sensitive to data corrup-

tions. This fact remains across the above important PCA classes in general and ST in particular. PCA dealing with impulsive noise and outliers is referred to as robust PCA. In 2011, it was revisited in a seminal work of Candes *et al* [42]. This work has attracted many research studies and applications, with over 4000 citations as of now. PCA for streaming data with impulsive noise and outliers is referred to as robust subspace tracking (RST). It is considered much more difficult than the original ST [43].

ST algorithms have been developed for over three decades [19, 20]. It has been around ten years since Delma's survey [20] and we thus believe it is not only important but the right time to do an up-to-date survey in order to highlight some aspects that were not mentioned in [20] as well as recent advances on this topic.

2.1.1 Related Work

Due to the importance of ST, there have been a number of published surveys in the literature. One of the first and earliest surveys on principal subspace tracking algorithms was carried on by Comon and Golub in [19]. The survey focuses on methods with high and moderate computational complexity for tracking the low-rank approximation of covariance matrices which may be slowly varying with time. In [20], Delmas provided a comprehensive overview on developments of classical ST algorithms with low (linear) complexity.

Recently, different adaptations of PCA for modern datasets and applications were reviewed in [44]. However, PCA for streaming data or ST was not addressed. The problem of tracking the underlying subspace of data from incomplete observations was discussed in [41] and [45]. Particularly, the former concerned methodological classes of ST algorithms that are able to deal with missing data while the latter presented a high-dimensional framework for analyzing their convergence behavior. The survey in [21] carried out reviews on robust PCA, RST, and robust subspace recovery in the presence of sparse outliers. Two similar surveys to [21] have also been conducted in [46] and [47] which respectively review (i) static and dynamic RPCA algorithms, and (ii) the entire body of works on robust sparse recovery. In the literature, there exist two others surveys on two adaptations of PCA which are distributed PCA [48] and sparse PCA [49]. The main contributions of the above-mentioned papers are summarized in Table 2.1.

2.1.2 Main Contributions

To the best of our knowledge, we are not aware of any work that reviews the RST problem in the presence of different kinds of non-Gaussian noise. Al-

Table 2.1: Surveys on PCA/SE and ST

Paper	Topic & scope	Main contribution
[19, 1990]	Principal ST	A survey on numerical methods for tracking the low-rank approximation of covariance matrices slowly varying with time.
[20, 2010]	Principal and minor ST	A comprehensive survey on classical ST algorithms.
[44, 2016]	Principal component analysis	A survey on adaptations of PCA for modern datasets and applications.
[45, 2018]	Principal ST	A high-dimensional analysis framework for the state-of-the-art ST algorithms from incomplete observations.
[41, 2018]	ST and streaming PCA	A survey on both classical and recent ST algorithms able to handle missing data and their performance guarantee.
[21, 2018]	Robust subspace learning	A survey on robust PCA, RST, and robust subspace recovery in the presence of sparse outliers.
[46, 2018]	Robust PCA	A survey on statistic and dynamic robust PCA algorithms.
[48, 2018]	Principal component analysis	A survey on distributed PCA algorithms.
[47, 2018]	Robust subspace recovery	A survey on works on robust subspace recovery when measurements are corrupted by sparse outliers.
[49, 2018]	Sparse PCA	A survey on recent theoretical developments of sparse PCA.
Ours	RST	A survey on RST algorithms in the presence of different kinds of corruptions (e.g. outliers, missing data, impulsive, and colored noise) and sparse subspace.

though the three surveys [21,46,47] reviewed some classes of RST algorithms, they only discussed on sparse outliers. Methods for other non-Gaussian noises (e.g., impulsive noise and colored noise) have not been reviewed yet. Moreover, no survey exists on the problem of sparse ST in the literature. This observation motivates us to carry out a survey on the topic.

The main goal of this survey is to fill the gap in the literature addressing the following three kinds of non-Gaussian noises (including outliers, impulsive noise, and colored noise) and sparse constraints. Our contributions are as follows. First, in the context of missing data and outliers, we review four main approaches for dealing with them. They are Grassmannian, recursive least-squares (RLS), recursive projected compressive sensing (ReProCS), and adaptive projected subgradient method (APSM). Second, when the measurements are corrupted by impulsive noise, we show that most of state-of-the-art RST algorithms are based on improving the well-known PAST algorithm which belongs to the class of RLS methods. Two other appealing approaches including weighted RLS and adaptive Kalman filtering are also reviewed. Third, we outline two main classes of RST algorithms that are able to deal with colored noise: instrumental variable-based and oblique projections. Finally, a short review on sparse ST algorithms is presented.

The structure of our review is as follows, please see Fig. 2.1 for an illustration. Section 2.2 states the problem of RST. In Section 2.3, we provide the state-of-the-art algorithms for the RST problem in the presence of missing data and outliers. The next two sections, 2.4 and 2.5, present RST algorithms that are able to handle impulsive noise and colored noise, respectively. Section 2.6 provides a short review on sparse ST. Finally, Section 2.7 concludes the chapter.

2.2   Robust Subspace Tracking: Problem Formulation

At each time t, we suppose to observe a signal $\mathbf{x}_t \in \mathbb{R}^n$ satisfying

$$\mathbf{x}_t = \mathbf{P}_t(\boldsymbol{\ell}_t + \mathbf{v}_t), \tag{2.1}$$

where $\mathbf{P}_t \in \mathbb{R}^{n \times n}$ is an observation mask matrix indicating the i-th entry of $\mathbf{x}_t$ is observed (i.e., $\mathbf{P}_t(i,i) = 1$) or not (i.e., $\mathbf{P}_t(i,i) = 0$), $\mathbf{v}_t \in \mathbb{R}^{n \times 1}$ is the (non-Gaussian) noise vector and $\boldsymbol{\ell}_t$ is the true signal living in a fixed or slowly time-varying low-dimensional subspace of $\mathbb{R}^n$. More concretely, $\boldsymbol{\ell}_t = \mathbf{U}_t \mathbf{w}_t$ in which $\mathbf{w}_t$ is a weight vector and $\mathbf{U}_t \in \mathbb{R}^{n \times r}$ ($r \ll n$) is a basis matrix with $d(\mathbf{U}_t, \mathbf{U}_{t-1}) \overset{\Delta}{=} \sin\theta(\mathbf{U}_t, \mathbf{U}_{t-1}) \ll 1$ where $\theta(\mathbf{U}_t, \mathbf{U}_{t-1})$ denotes the largest principal angle between $\mathbf{U}_t$ and $\mathbf{U}_{t-1}$. The RST problem can be stated as follows:
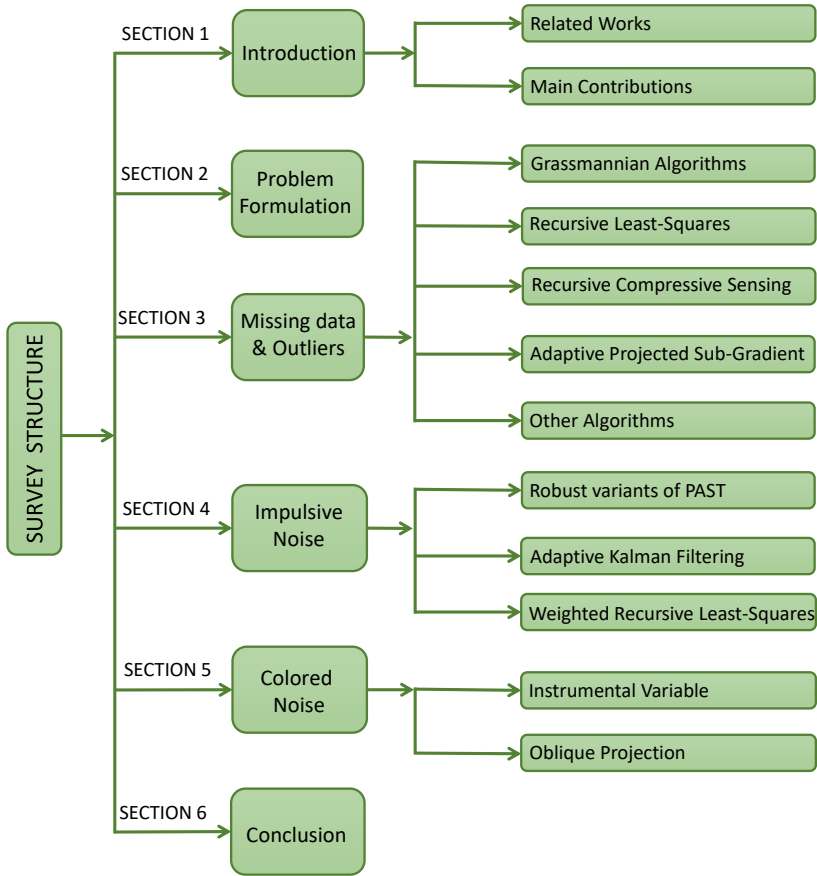
Figure 2.1: The structure of the survey.

Robust Subspace Tracking: *Given a streaming set of observed signals* $\{\mathbf{x}_t\}_{t \geq 1}$ *in (3.1), we wish to estimate a rank-r matrix* $\mathbf{U}_t$ *such that it can cover the span of the complete-data noiseless signal* $\boldsymbol{\ell}_t$.

In this chapter, we consider the RST problem in the presence of different kinds of the non-Gaussian noise $\mathbf{v}_t$: sparse outliers, impulse noise, and colored noise. Also, we review sparse ST algorithms under the constraint that the basis matrix $\mathbf{U}_t$ is sparse.

2.3   Robust Subspace Tracking in the Presence of Missing Data and Outliers

In the literature, there have been several studies on ST in the presence of outliers and missing data. The proposed RST algorithms can be categorized into four main classes: (i) Grassmannian, (ii) recursive east-Squares (RLS), (iii) recursive projected compressive sensing (ReProCS), and (iv) adaptive projected subgradient method (APSM). We summarize all the RST algorithms robust to outliers and missing data in Table 2.2.

2.3.1   Grassmannian Algorithms

Many of RST algorithms are based on the Grassmannian approach in which the ST procedure can be cast into an optimization process on a Grassmann manifold. More concretely, Grassman manifold is a space that parameterizes all r-dimensional linear subspaces of the N-dimensional vector space. The underlying subspace can be derived from averaging the column span of the (fully or partially) observed signals on the Grassmannian. Interestingly, each observed signal ℓ_t spans a one-dimensional subspace which can be described as a point in the Grassmannian. Therefore, the Grassmannian approach offers several advantages such as a lower number of parameters to optimize and limited memory usage and the resulting RST algorithms are often efficient and scalable to high dimensional data [71].

The state-of-the-art RST algorithms include GRASTA [50], GOSUS [51], pROST [52, 53], and RoIGA [68, 69]. In [50], He *et al.* proposed an efficient RST algorithm called Grassmannian robust adaptive ST (GRASTA) which is a robust version of GROUSE in [72]. GRASTA first uses an ℓ_1-norm cost function to reduce the effect of sparse outliers and then performs the incremental gradient on the Grassmann manifold of the subspace $\mathbf{U}$. In [51], Xu *et al.* introduced an effective algorithm namely GOSUS for tracking subspace with structured-sparsity. GOSUS also incorporates an adaptive step-size for the incremental gradient on the manifold. The effectiveness of GOSUS was demonstrated via the real application of video background subtraction and multiple face tracking. In [52, 53], Hage *et al.* proposed a method, namely pPOST that combines the advantages of Grassmannian optimization with a non-convex sparsity measure. Instead of using the ℓ_1-norm regularization, pPOST uses the penalty with non-convex ℓ_0-surrogates allows reconstruction even in the case when ℓ_1-based methods fail. Another algorithm dubbed robust intrinsic Grassmann average (RoIGA) was proposed by Rudrasis *et al.* in [68, 69]. RoIGA is a geometric approach to computing principal linear subspaces in finite and infinite dimensional reproducing kernel Hilbert spaces.

Table 2.2: Robust subspace tracking algorithms in the presence of both missing data and sparse outliers.

Algorithm	Approach	Missing Data?	Sparse Outliers?	Prior Information	Warm Start?	Convergence Guarantee	Computational Complexity
GRASTA (2012 [50])	ℓ_1-norm + ADMM + Grassmannian	✓	✓	✗	random	✗	$O(nr + r^3)$
GOSUS (2014 [51])	ℓ_2-norm + ADMM + Grassmannian	✗	✓	✓	random	✗	-
pROST (2014 [52,53])	ℓ_0-norm + Grassmannian + Conjugate Gradient	✗	✓	✗	random	✗	-
MRMD (2014 [54])	Online max-norm regularization	✗	✓	✓	random	✓	-
ROSETA (2015 [55])	$\ell_{1,2}$-norm + ADMM + RLS	✓	✓	✗	random	✗	$O(nr^2)$
Roubst STAPSM (2015 [56,57])	APSM + CoSAMP*	✓	✓	✗	random	✓	$O(knr^2)$
ReProCS-cPCA (2016 [58])	ReProCS	✗	✓	✓	batch	✓	$O(nr\log^2(n)\log(1/\epsilon))°$
OTNNR (2016 [59])	Truncated nuclear-norm regularization	✗	✓	✗	random	✗	-
OLP-RPCA (2017 [60])	ℓ_p-norm + singular value thresholding	✗	✓	✗	random	✓	$O(nr + r^3)$
L1-PCA (2018 [61])	ℓ_1-norm + Bit-flipping	✗	✓	✗	batch	✗	$O(nr\omega^2)‡$
PETRELS-CFAR (2018 [62])	Robust statistic + RLS	✓	✓	✓	batch	✗	$O(nr^2 + n\omega)^+$
s-ReProCS (2019 [63])	ReProCS	✓	✓	✓	batch	✓	$O(nr\log(n)\log(1/\epsilon))°$
NORST-miss (2019 [64])	ReProCS	✓	✗	✓	batch	✓	$O(nr\log(1/\epsilon))°$
L1-IRW (2019 [65])	ℓ_1-norm + Bit-flipping	✗	✓	✗	batch	✗	$O(k(nwr^3p + 2^r nr^2))^†$
OSTP (2019 [66])	Schatten quasi-norm + Block-proximal gradient	✗	✓	✗	random	✓	$O(nr^2)$
NORST (2020 [67])	ReProCS	✓	✓	✓	batch	✓	$O(nr\log(1/\epsilon))°$
RoIGA (2020 [68,69])	IGA# + Grassmannian	✗	✓	✗	random	✗	-
PETRELS-ADMM (2021 [25,70])	ℓ_1-norm + ADMM + RLS	✓	✓	✓	random	✓	$O(nr^2)$

IGA: Intrinsic Grassmann Average
* CoSAMP: Compressed Sampling Orthogonal Matching Pursuit
+ ω: length of training window
° ϵ : a desired subspace recovery accuracy
‡ ω: length of sliding window
† ω: length of sliding window, k: number of iterations, and p: number of bit flips

Among them, RoIGA is shown as one of the fastest RST algorithms for handling missing data corrupted by outliers.

2.3.2 Recursive Least-Squares based Algorithms

Another line of the RST research is based on recursive least-squares (RLS) methods where the underlying subspace is recursively updated by minimizing a (weighted) least-squares objective function containing squared residuals and a penalty accounting for outliers. An efficient RLS-based algorithm is parallel estimation and tracking by recursive least squares (PETRELS) [73] which can be considered as an extension of the projection approximation ST (PAST) algorithm [74] in order to handle missing data.

Inspired by PETRELS, several robust variants have been proposed to deal with outliers the same line such as [25, 55, 62, 70]. Robust online subspace estimation and tracking (ROSETA) in [55] applies an adaptive step size at the stage of subspace estimation to enhance the convergence rate. Meanwhile the main idea of PETRELS-CFAR algorithm [62] is to handle "outliers-removed" data (i.e., outliers are first removed before performing ST) using a Constant False Alarm Rate (CFAR) detector. Adopting the approach of PETRELS-CFAR, but aiming to improve RST performance, we proposed an efficient algorithm called PETRELS-ADMM which is able to remove outliers more effectively in [25,70]. It includes two main stages: outlier rejection and subspace estimation and tracking. Outliers living in the measurement data are detected and removed by a ADMM solver in an effective way. An improved PETRELS was then introduced to update the underlying subspace. In practice, the convergence rate of RST-type algorithms is often faster than that of Grassmmannian-based algorithms in slowly time-varying environments.

2.3.3 Recursive Projected Compressive Sensing based Algorithms

Recursive projected compressive sensing (ReProCS)-based algorithms [58,63, 64, 67] are also capable of tracking subspace in the presence of outliers and missing data.

ReProCS-type algorithms use the piecewise constant subspace change model described previously and start with a "good" estimate of the initial subspace. At each time, they first solve a projected compressive sensing problem to derive the sparse outliers, e.g., using ℓ_1 minimization followed by thresholding-based support estimation. After that, the subspace direction change is then estimated by using projection-SVD [63].

ReProCS provides not only a memory-efficient and highly robust solution, but also a precise subspace estimation compared to the state-of-the-arts. However, ReProCS-type algorithms often require strong assumptions on subspace changes, outlier magnitudes, and accurate initialization.

2.3.4 Adaptive Projected Subgradient Method based Algorithms

Adaptive projected subgradient method (APSM) can provide a robust solution to the presence of missing data and outliers [56, 57]. Main advantages of APSM are that convex constraints can be readily incorporated and it can be used as an alternative to constructing the cost function from the sum of square errors like RLS methods. The key idea of APSM stems from that unknown parameters of regression models can be estimated from seeking a point in the intersection of all the sets defined by measurements. In the context of ST, based on the latest observed signals, a cost function is properly chosen at each time instant which scores a zero loss. The next task is to reach the intersection point. To deal with sparse outliers, APSM-type algorithms detect the time instances at which the observed signals are corrupted by outliers via using sparsity-aware greedy techniques (e.g. compressed sampling orthogonal matching pursuit as used in [57]) and then reject them.

2.3.5 Other Algorithms

Some other RST algorithms are able to track the underlying subspace over time from measurements corrupted by sparse outliers such as MRMD [54], OTNNR [59], L1-PCA [61], L1-IRW [65], OLP-RPCA [60], and OSTP [66]. Most of them use a ℓ_p-regularization ($0 \leq p \leq 1$) to discard the effect of outliers. However, they are not designed for missing data.

2.4 Robust Subspace Tracking in the Presence of Impulsive Noise

By "impulsive", we mean it can be burst noise [75, 76], spherically invariant random variable (SIRV) noise [77, 78], or alpha-stable noise [79, 80]. We note that even though these algorithms were described to reduce the effect of impulsive noise in general, most simulation results were shown for burst noise only. RST algorithms that are robust to impulsive noise are summarized in Table 2.3.

2.4.1 Robust Variants of PAST

To take into account impulsive noise, some methods proposed in the literature have mainly been based on robust statistics so far. Among them, some studies have proposed robust variants of PAST to deal with impulsive noise. In [81], a robust PAST (RPAST) was proposed. The algorithm first detects the

Table 2.3: Robust subspace tracking algorithms in the presence of impulsive noise.

Algorithm	Approach	Burst noise	SIRV noise	α-stable noise	Warm Start?	Convergence Guarantee	Computational Complexity
RPAST (2006 [81])	PAST + M-estimation	✓	-	✓	random	✓	$O(nr + r^2)$
MCC-PAST (2014 [82])	Maximum correntropy criterion (MCC) + PAST	✓	-	✓	random	✗	$O(nr + r^2)$
BNC-PAST (2014 [83])	Bounded nonlinear covariance (BNC) + PAST	✓	-	✓	random	✗	$O(nr + r^2)$
robust KFVM (2020 [84])	Adaptive Kalman filter + M-estimation	✓	-	-	random	✗	$O(nr\ell + \ell r^2) + O(\ell^2 r + \ell^3)$
ROBUSTA (2018 [62])	Weighted RLS + Mahalanobis distance	✓	✓	✓	random	✓	$O(nr + r^2)$

ℓ: length of the sliding window

$-$: unknown or undetermined

occurrence of the impulsive noise based on a threshold, and then eliminates undesirable effects by discarding contaminated observations. The threshold is determined based on an empirical function of noise variance with the assumption that error vectors follow a Gaussian distribution corrupted by additive impulsive noise.

Zhang *et al.* introduced another PAST's variant called MCC-PAST via the maximum correntropy criterion (MCC) in [82, 85, 86]. MCC-PAST exploits a correntropy as a new statistic, which can quantify both the time structures and statistics of two random processes, to deal with impulsive noise. Accordingly, the maximum correntropy criterion (MCC) is applied as a substitute for the mean square error criterion in the objective function of PAST. Based on the RLS technique, the MCC-PAST algorithm was then developed. To extend the tracking capability of the MCC-PAS, a variable forgetting factor (FF) technique was also employed in the recursion process. In parallel, Shengyang *et al.* developed another robust variant of PAST, namely BNC-PAST, to track the underlying subspace via a different criterion [83]. The authors defined a new concept namely bounded non-linear covariance (BNC) to handle relative problems (including ST) in the presence of non-Gaussian noise with a heavy-tailed distribution. In particular, bounded nonlinear maps were employed to discard the effect of impulsive noise. Accordingly, a new robust PAST algorithm based on BNC was derived.

2.4.2 Adaptive Kalman Filtering

Another good approach capable of handling impulsive noise is based on adap-

tive Kalaman filtering. In [84], Liao *et al.* proposed a RST algorithm based on an adaptive Kalman filter with variable number of measurements (KFVM). The main idea of using the KFVM is to deal with the tracking of fast-varying subspace [87]. More concretely, when the underlying subspace varies quickly, a small number of past observations are exploited in the recursion and vice versa. To handle the impulsive noise, the M-estimate technique is incorporated into the KFVNM algorithm. The complexity of the proposed KFVM-based algorithm is much higher than the PAST-based algorithms especially when the number of observations used for subspace update is large.

2.4.3 Weighted Recursive Least-Squares Method

Recently, based on robust statistics but different from the common two-step scheme mentioned above, we proposed in [62] an RST algorithm with linear computational complexity based on a weighted RLS approach, namely ROBUSTA. On the theoretical aspect, we provided a converge analysis of ROBUSTA in the presence of SIRV noise. Interestingly, we showed that it also corresponded to adaptive robust covariance estimation. ROBUSTA outperformed many state-of-the-art algorithms for burst noise, SIRV noise, and alpha-stable noise. Also, it can be easily adapted, in conjunction with preprocessing steps, to handle alpha-stable noise.

2.5 Robust Subspace Tracking in the Presence of Colored Noise

In the literature, RST algorithms that are robust to colored noise can be categorized into two groups: (i) instrumental variable and (ii) oblique projection. We summarize these algorithms in Table 2.5.

2.5.1 Instrumental Variable based Algorithms

For colored noise, one of the main directions is to use the instrumental variable (IV) which allows avoiding biased estimate. An appealing benefit of this approach is easy to adapt derivation from classical ST algorithms. While having improved performance, the computational complexity of IV-based algorithms is often higher than the original ones due to the selection of the IV vector size. Specifically, in [88], two direct extensions of the PAST algorithms, named IV-PAST and extended IV-PAST, were proposed. It is shown that their performance is enhanced, comparing to the original ones. With the aim to improve further performance in subspace-based system identification applications, several algorithms in conjunction with using IV were addressed

Table 2.4: Robust subspace tracking algorithms in the presence of colored noise.

Algorithm	Approach	Warm Start?	Guarantee Convergence	Computational Complexity
IV-PAST (2012 [88])	IV + PAST	random	✗	$3n\ell + O(nr)$
IVPM (2014 [89])	IV + propagator-based	random	✗	$n(\ell + 2r)$
LOFF-VR-SREIV-PAST (2020 [90])	IV + PAST + adaptive forgetting factor	random	✓	$6nr + 5r^2 + 4n + 14r + O(nr)$
obPAST (2005 [91])	Oblique projection + PAST	random	✗	$3nr^2 + 3nr + O(r^3)$
obYAST (2012 [92])	Oblique projection + YAST	random	✗	$5nr + O(r^2 + n) + O(r^3)$

ℓ: the dimension of instrumental variable (IV) vector.

in [89]. The key idea is to adapt the propagator approach by exploiting the relationship between array signal processing and subspace identification.

Very recently, Chan *et al.* in [90] proposed a new robust variant of PAST capable of handing linear models with complex coefficients, multiple outputs, and colored noises. In the proposed method, the authors used a new adaptive forgetting factor and imposed a ℓ_2-norm regularization into the objective function of PAST. In particular, the adaptive forgetting factor was obtained at each time instant by minimizing the mean-square deviation of the estimator from an extended IV linear model and IV-PAST. The additional ℓ_2-norm regularized term on the weight vectors is aimed to reduce the error variance and prevent the ill-conditioned computation at low SNR levels. Generally, if low computational complexity is concerned, IV-based methods require a IV vector uncorrelated with the noise which is not always met in practice.

2.5.2 Oblique Projection based Algorithms

Another direction, which can avoid the above drawback, is based on oblique projection onto the subspace manifold, such as [91,92]. It is due to the fact that the noise vector may lie in a low dimension subspace instead of being treated as full rank in the observation space. Naturally, oblique projections arise in the solution to recover the signal. Accordingly, Chen *et al.* proposed a variant of PAST named oblique PAST (obPAST) to track the signal subspace in [91]. In the same line, based on the well-known YAST algorithm [99], Florian *et al.* introduced the new obYAST algorithm in [92]. Both obPAST and obYAST minimized a new exponential least-squares cost function where the orthog-

Table 2.5: Sparse subspace tracking algorithms

Algorithm	Approach	Prior Information	Warm Start?	Guarantee Convergence	Computational Complexity
OIST (2016 [93])	Oja method + Soft-thresholding	✗	random	✓	$O(nr)$
Streaming SPCA (2015 [94])	Row truncation + QR decomposition	✗	batch	✓	$O(nr \min(r, s \log n))$
ℓ_1-PAST (2016 [95])	PAST method + ℓ_1-norm sample matrix inverse	✓	random	✗	$3nr^2 + 3nr + O(r^2)$
OVBSL (2017 [96])	Bayesian inference + ℓ_2/ℓ_1-norm promotion	✓	random	✗	$O(nr^2 + nr)$
SS/DS-OPAST (2017 [97])	2-step approach + OPAST + ℓ_1-norm approximation	✗	random	✗	$3nr^2 + 3nr + O(r^3)$/ $3nr + O(nr^2)$
SS/GSS-FAPI (2020 [98])	2-step approach + FAPI + Givens rotations	✗	random	✓	$2nr^2 + 4nr + O(r^2)$/ $4nr + 4ns + O(r^2)$

onal projection in the residual error term is replaced with an oblique one. Experiment results indicate that this modification can facilitate the tracking ability of PAST and YAST in the presence of colored noise. Table 2.4 reports further information about these RST algorithms, e.g., convergence and computational complexity.

2.6   Sparse Subspace Tracking

Recently, sparse subspace estimation and tracking have been attracted more attention from the signal processing community due to the fact that many modern datasets admit sparse representation has huge potential capabilities for analyzing them [100]. Although several algorithms have been introduced for sparse subspace estimation in the batch setting (see [101–103] for examples), there exist only a few studies on sparse ST algorithms so far.

In [93], Chuang and Yue proposed an adaptive algorithm called OIST (which stands for Oja's algorithm with Iterative Soft Thresholding) for online sparse PCA. The authors investigated a rank-one spiked model in a high-dimension regime and indicated that the estimate of the eigenvector from the sample covariance matrix is inconsistent. To alleviate it, they introduced an extended version of Oja's algorithm followed by a soft-thresholding step to promote sparsity on the estimate. The asymptotic convergence, steady state, and phase transition of OIST were also derived to understand its behavior in a high-dimension regime when the dimension is much larger than the number of observations. However, OIST is designed for only rank-one subspaces, i.e. lines. In parallel, a novel online sparse PCA algorithm able to deal with rank-

k spiked models ($k \geq 1$) was proposed via row truncation technique in [94]. More concretely, a simple ℓ_2-norm based row truncation operator was introduced to zero out rows whose leverage score is below a predefined threshold. At each time instant, the QR decomposition of the resulting truncated covariance matrix was realized to update the principal subspace. The authors also proved that the proposed algorithm is consistent in the high-dimension regime.

In [95], Xiaopeng *et al.* introduced a new robust variant of PAST called ℓ_1-PAST. Specifically, the authors modified the cost function of PAST by adding a ℓ_1-norm constraint imposed on the subspace matrix to control its sparsity. Accordingly, a new RLS algorithm like PAST was derived to minimize the proposed objective function in an efficient way. The ℓ_1-PAST is robust and stable even when the number of observations is small.

In [96], Giampouras *et al.* developed a novel robust sparse ST method namely OVBSL in the lens of Bayesian inference. To deal with the sparsity constraint on the subspace matrix, OVBSL utilized the group-sparsity inducing the convex ℓ_2/ℓ_1-norm. Since it belongs to the family of Bayesian methods, no fine-tuning parameter is required and the proposed algorithm is fully automated.

In this topic, we also proposed several two-stage approach based algorithms for sparse ST in [97,98,104]. The main steps of the two-stage approach is as follows. We first utilize a well-known ST algorithm from the literature (e.g. PAST or FAPI) to extract an orthonormal basis of the underlying subspace. Then, we estimate a sparse weight matrix based on some criteria on sparsity such that it can span the same subspace. For example, in [97], two new algorithms SS-OPAST and DS-OPAST were designed for sparse system matrix and sparse source signals respectively. We particularly exploited the natural gradient to find the sparsest matrix from the estimated orthonormal matrix by OPAST. In [98, 104], we used FAPI in the first stage and then derived SS-FAPI, orthogonal SS-FAPI, and GSS-FAPI algorithms. Specifically, the sparsity criterion considered there is differentiable and smoother than the previous one in [97]. Accordingly, it facilitates the optimization by employing the Newton method and Taylor expansions. To sum up, a performance comparison among these sparse ST algorithms is given in Table 2.5.

2.7 Conclusions

ST has shown an increased interest in signal processing with the aim of analysing real-time big data problems and its improvement is in parallel to recent advances in optimization. In this chapter, we provided a brief survey on adaptive algorithms for RST which were mostly developed over the last decade. We

highlighted three classes of RST algorithms for dealing with non-Gaussian noises including sparse outliers, impulsive noise, and colored noise. The last decade has also witnessed the widespread of high-dimensional data analysis in which sparse representation-based methods are successfully applied to many signal processing applications. Accordingly, sparse ST algorithms are also reviewed in this chapter.

Robust Subspace Tracking with Missing Data and Outliers

3

In this chapter, we propose a novel algorithm, namely PETRELS-ADMM, to deal with subspace tracking in the presence of sparse outliers and missing data. The proposed approach consists of two main stages: outlier rejection and subspace estimation. In the first stage, alternating direction method of multipliers (ADMM) is effectively exploited to detect outliers affecting the observed data. In the second stage, we propose an improved version of the parallel estimation and tracking by recursive least squares (PETRELS) algorithm to update the underlying subspace in the missing data context. We then present a theoretical convergence analysis of PETRELS-ADMM which shows that it generates a sequence of subspace solutions converging to the optimum of its batch counterpart. The effectiveness of the proposed algorithm, as compared to state-of-the-art algorithms, is illustrated on both simulated and real data.

3.1   Introduction

Subspace estimation plays an important role in signal processing with numerous applications in wireless communications, radar, navigation, image/video processing, biomedical imaging, etc. [105], especially processing modern data in today's big and messy data [43]. It corresponds to estimating an appropriate r-dimensional subspace $\mathbf{U}$ of $\mathbb{R}^n$ where $r \ll n$, from a set of m observed data vectors $\{\mathbf{x}_i\}_{i=1}^m$, or equivalently, a measurement data matrix $\mathbf{X}$ of size $n \times m$. To this end, the standard approach is to solve an eigen-problem in a batch manner where the underlying subspace can be obtained from either singular value decomposition of the data matrix or eigenvalue decomposition of its covariance matrix. In certain online or large-scale applications, batch algorithms become inefficient due to their high computational complexity, $O(nm \min(m,n))$, and memory cost, $O(nm)$ [9].

In the signal processing literature, several good surveys of the standard algorithms for subspace tracking can be found, e.g., [19, 105]. The algorithms can be categorized into three classes in terms of their computational complexity: high complexity $O(n^2 r)$, moderate complexity $O(nr^2)$ and low complexity $O(nr)$. Note that, there usually exists a trade-off among estimation accuracy, convergence rate and computational complexity. However, the performance of standard algorithms may be degraded significantly if the measurement data are corrupted by even a small number of outliers or missing observations [44]. Recent surveys [21, 41, 45] show that missing data and outliers are ubiquitous and more and more common in the big data regime. This has led to attempts to define robust variants of subspace learning, namely robust subspace tracking (RST), or online robust PCA. In this work, we aim to investigate the RST problem in the presence of both sparse outliers and missing data.

Our study is also motivated by several emerging applications in diverse fields. In big data analysis, subspace tracking is used to monitor dynamic cardiac magnetic resonance imaging (MRI), track network-traffic anomalies [106] or mitigate radio frequency interference (RFI) in radio astronomy [107]. Moreover, in 5G wireless communication, subspace tracking have recently been exploited for channel estimation in massive MIMO [108] and millimeter wave multiuser MIMO [109].

3.1.1 Related Works

In the literature, there have been several studies on subspace tracking in the missing data context. Among them, Grassmannian rank-one update subspace estimation (GROUSE) [72] is an incremental gradient subspace algorithm that performs the stochastic gradient descent on the Grassmannian manifold of the r-dimensional subspace. It belongs to the class of low complexity and its convergence has recently been proved in [110]. A robust version of GROUSE for handling outliers is Grassmannian robust adaptive subspace tracking (GRASTA) [50]. GRASTA first uses an ℓ_1-norm cost function to reduce the effect of sparse outliers and then performs the incremental gradient on the Grassmannian manifold of subspace $\mathbf{U}$ in a similar way as in GROUSE. Although GRASTA is one of the fastest RST algorithms for handling missing data corrupted by outliers, convergence analysis of this algorithm is not available.

Parallel estimation and tracking by recursive least squares (PETRELS) [73] can be considered as an extension of the well-known projection approximation subspace tracking (PAST) algorithm [74] in order to handle missing data. Specifically, PETRELS is a recursive least squares-type algorithm applying the second order stochastic gradient descent to the cost function. Inspired by PETRELS, several variants have been proposed to deal with missing data in the same line such as [55, 62, 106]. The subspace tracking algorithm in [106] is derived from minimizing the sum of squared residuals, but adding a regularization of the nuclear norm of subspace $\mathbf{U}$. Robust online subspace estimation and tracking (ROSETA) in [55] applies an adaptive step size at the stage of subspace estimation to enhance the convergence rate. Meanwhile the main idea of PETRELS-CFAR algorithm [62] is to handle "outliers-removed" data (i.e., outliers are first removed before performing subspace tracking) using a constant false alarm rate (CFAR) detector. However, the convergence of these PETRELS-based algorithms has not been mathematically proved yet.

Recursive projected compressive sensing (ReProCS)-based algorithms [63, 64] are also able to adaptively reconstruct a subspace from missing observations. They provide not only a memory-efficient solution, but also a precise subspace estimation as compared to the state-of-the-arts. However, they re-

quire strong assumptions on subspace changes, outlier magnitudes and accurate initialization.

Other subspace tracking algorithms, able to deal with missing data, include pROST [53], APSM [57], POPCA [111] and OVBSL [96]. They either require memorizing previous observations and good initialization or do not provide a convergence guarantee.

Among the subspace tracking algorithms reviewed above, only a few of them are robust in the presence of both outliers and missing observations, including GRASTA [50], pROST [53], ROSETA [55], ReProCS-based algorithms [63, 64] and PETRELS-CFAR [62].

3.1.2 Contributions

Adopting the approach of PETRELS-CFAR [62] but aiming to improve RST performance, we are interested in looking for a method that can remove outliers more effectively. Following our preliminary study presented in [70], the main contributions of the chapter are as follows.

First, we propose a novel algorithm, namely PETRELS-ADMM, for the RST problem to deal with both missing data and outliers. It includes two main stages: outlier rejection and subspace estimation and tracking. Outliers residing in the measurement data are detected and removed by our ADMM solver in an effective way. Particularly, we design an efficient augmented Lagrangian alternating direction method for the ℓ_1-regularized loss minimization. Furthermore, we propose an improved version of PETRELS, namely iPETRELS. It is observed that PETRELS is ineffective when the fraction of missing data is too large. We thus add a regularization of the $\ell_{2,\infty}$-norm, which aims to control the maximum ℓ_2-norm of rows in $\mathbf{U}$, in the objective function to avoid such performance loss. In addition, we introduce an adaptive step size to speed up the convergence rate as well as enhance the subspace estimation accuracy.

Second, we provide a convergence analysis of the proposed algorithm where we show that the solutions $\{\mathbf{U}_t\}_{t=1}^{\infty}$ generated by PETRELS-ADMM converge to a stationary point of the expected loss function $f(\mathbf{U})$ asymptotically. To the best of our knowledge, this is a pioneer analysis for RST algorithm's convergence in the presence of both outliers and missing data, *under mild conditions*.

Finally, we provide extensive experiments on both simulated and real data to illustrate the effectiveness of PETRELS-ADMM in three application contexts: robust subspace tracking, robust matrix completion and video background foreground separation.

There are several differences between PETRELS-ADMM and the state-of-the-art RST algorithms. In particular, our mechanism for outlier rejection can facilitate the subspace estimation ability of RST algorithms where "clean"

data involve the process only, thus improving overall performance. Excepting PETRELS-CFAR, the common principle of the state-of-the-art algorithms is "outlier-resistant" (i.e., to have a "right" direction toward the true subspace). The algorithms thus require robust cost functions as well as additional adaptive parameter selection. For examples, GRASTA and ROSETA use the ℓ_1-norm robust estimator to reduce the effect of outliers while pROST applies the ℓ_0-norm one instead. However, there is no guarantee that the ℓ_p-norm robust estimator (i.e., $p \in [0, 1]$) can provide an optimal solution because of non-convexity. Accordingly, the effect of outliers can not be completely removed in tracking. This is why the algorithms can fail in the appearance of a large fractions of outliers or significant subspace changes in practice. By contrast, "detect and skip" approach like PETRELS-CFAR can utilize advantage (i.e., competitive performance) of the original PETRELS in missing observations and then treat outliers as missing data to facilitate the subspace tracking.

Compared to PETRELS-CFAR, our ADMM solver may be efficient than CFAR in terms of memory cost and flexibility. The constant false alarm rate method (CFAR) [112] uses a moving window to detect outliers (i.e., using both old and new observations at each time instant). By contrast, our ADMM solver exploits only a new incoming data vector, hence requiring a lower storage complexity. Moreover, the performance of CFAR depends on predefined parameters such as the probability of false alarm and the size of the reference window [62]. Our ADMM solver does not involve such parameters and hence it is more efficient. Third, PETRELS-CFAR may provide an unstable solution in the presence of a high corruption fraction due to lack of regularization (i.e., in the similar way as PETRELS).

Moreover, PETRELS-ADMM can be classified to a class of provable ST algorithms [63,64] where a performance guarantee is provided. Our proposed algorithm takes both advantages of streaming solution (need only single-pass of data) and preserved convergence.

The structure of the chapter is organized as follows. Section 3.2 formulate the RST problem. Section 3.3 establishes our PETRELS-ADMM algorithm for RST and Section 3.4 gives its theoretical convergence analysis. Section 3.5 presents extensive experiments to illustrate the effectiveness of PETRELS-ADMM as compared to the state-of-the-art algorithms. Section 3.6 concludes the chapter.

3.2 Problem Formulation

Assume that at each time t, we observe a signal $\mathbf{x}_t \in \mathbb{R}^n$ satisfying the following model:

$$\mathbf{x}_t = \mathbf{P}_t(\boldsymbol{\ell}_t + \mathbf{n}_t + \mathbf{s}_t), \tag{3.1}$$

where $\boldsymbol{\ell}_t \in \mathbb{R}^n$ is the true signal that lies in a low dimensional subspace[1] of $\mathbf{U} \in \mathbb{R}^{n \times r}$ (i.e., $\boldsymbol{\ell}_t = \mathbf{U}\mathbf{w}_t$, where $\mathbf{w}_t$ is a weight vector and $r \ll n$), $\mathbf{n}_t \in \mathbb{R}^n$ is the noise vector, $\mathbf{s}_t \in \mathbb{R}^n$ is the sparse outlier vector, while the diagonal matrix $\mathbf{P}_t \in \mathbb{R}^{n \times n}$ is the observation mask indicating whether the k-th entry of $\mathbf{x}_t$ is observed (i.e., $\mathbf{P}_t(k, k) = 1$) or not (i.e., $\mathbf{P}_t(k, k) = 0$). For the sake of convenience, let Ω_t be the set of observed entries at time t.

Before introducing the RST formulation, we first define a loss function $\ell(.)$ that remains convex while still promoting sparsity: For a fixed subspace $\mathbf{U} \in \mathbb{R}^{n \times r}$ and a signal $\mathbf{x} \in \mathbb{R}^n$ under an observation mask $\mathbf{P}$, the loss function $\ell(\mathbf{U}, \mathbf{P}, \mathbf{x})$ with respect to $\mathbf{U}$ and $\{\mathbf{P}, \mathbf{x}\}$ is derived from minimizing the projection residual on the observed entries and accounting for outliers as

$$\ell(\mathbf{U}, \mathbf{P}, \mathbf{x}) \overset{\Delta}{=} \min_{\mathbf{s}, \mathbf{w}} \; \tilde{\ell}(\mathbf{U}, \mathbf{P}, \mathbf{x}, \mathbf{w}, \mathbf{s}), \tag{3.2}$$

$$\text{with } \tilde{\ell}(\mathbf{U}, \mathbf{P}, \mathbf{x}, \mathbf{w}, \mathbf{s}) = \left\| \mathbf{P}(\mathbf{U}\mathbf{w} + \mathbf{s} - \mathbf{x}) \right\|_2^2 + \rho \left\| \mathbf{s} \right\|_1, \tag{3.3}$$

where we here use the ℓ_1 regularization to promote entry-wise sparsity on $\mathbf{s}$ and $\rho > 0$ is a regularization parameter to control the degree of the sparsity.[2]

Now, given a streaming set of observed signals, $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^t$ in (3.1), we wish to estimate a rank-r matrix $\mathbf{U}_t \in \mathbb{R}^{n \times r}$ such that it can cover the span of the complete-data noiseless signal $\boldsymbol{\ell}_t$. RST can be achieved via the following minimization problem:

$$\mathbf{U}_t = \underset{\mathbf{U} \in \mathbb{R}^{n \times r}}{\operatorname{argmin}} \left[f_t(\mathbf{U}) \overset{\Delta}{=} \frac{1}{t} \sum_{i=1}^t \beta_i^{t-i} \ell(\mathbf{U}, \mathbf{P}_i, \mathbf{x}_i) \right], \tag{3.4}$$

where the forgetting factor $\beta_i \in (0, 1]$ is to discount the effect of past observations. For the convergence analysis, we will consider the expected cost $f(\mathbf{U})$ on signals distributed by the true data-generating distribution $\mathbb{P}_{\text{data}}$, instead of the empirical cost $f_t(\mathbf{U})$. Thanks to the law of large numbers, expectation

---

[1] In an adaptive scheme, this subspace might be slowly time-varying, i.e., $\mathbf{U} = \mathbf{U}_t$, and hence the adaptive RST algorithm introduced next would not only estimate $\mathbf{U}$ but also track its variations along the iterations.

[2] The most direct way of enforcing sparsity constraints is to control the ℓ_0-norm of the solution which counts the number of non-zero entries. Following this way, the problem of (3.2) is well specified but computationally intractable. Interestingly, the ℓ_1 relaxation can recover the original sparse solution of the ℓ_0 problem while still preserving convexity [113].

of the observations without discounting (i.e., $\beta = 1$) converges to the true value when t tends to infinity,

$$\hat{\mathbf{U}} = \underset{\mathbf{U} \in \mathbb{R}^{n \times r}}{\text{argmin}} \left[f(\mathbf{U}) \stackrel{\Delta}{=} \mathbb{E}_{\mathbf{x} \stackrel{\text{i.i.d}}{\sim} \mathbb{P}_{\text{data}}} [\ell(\mathbf{U}, \mathbf{P}, \mathbf{x})] = \lim_{t \to \infty} f_t(\mathbf{U}) \right]. \tag{3.5}$$

From the past estimations $\{\mathbf{s}_i, \mathbf{w}_i\}_{i=1}^t$, instead of minimizing the empirical cost function $f_t(\mathbf{U})$ in (3.4), we propose to optimize the surrogate $g_t(\mathbf{U})$ of $f_t(\mathbf{U})$, which is defined as

$$g_t(\mathbf{U}) = \frac{1}{t} \sum_{i=1}^t \beta_i^{t-i} \left(\left\| \mathbf{P}_i (\mathbf{U}\mathbf{w}_i + \mathbf{s}_i - \mathbf{x}_i) \right\|_2^2 + \rho \left\| \mathbf{s}_i \right\|_1 \right), \tag{3.6}$$

where $\{\mathbf{s}_i, \mathbf{w}_i\}_{i=1}^t$ are considered as constants. Note that, the surrogate function provides an upper bound on $f_t(\mathbf{U})$. In our convergence analysis, we will prove that $f_t(\mathbf{U}_t)$ and $g_t(\mathbf{U}_t)$ converge almost surely to the same limit. As a result, the solution $\mathbf{U}_t$ obtained by minimizing $g_t(\mathbf{U})$ is exactly the solution of $f_t(\mathbf{U})$ when t tends to infinity.

3.2.2   Assumptions

We make the following assumptions for convenience of convergence analysis as well as helping deploy our optimization algorithm:

(A-1): The data-generation distribution $\mathbb{P}_{\text{data}}$ has a compact support, $\mathbf{x} \stackrel{\text{i.i.d}}{\sim} \mathbb{P}_{\text{data}}$. Indeed, real data are often bounded such as audio, image and video, hence this assumption naturally holds in many situations.

(A-2): $\mathbf{U}$ is constrained to the set $\mathcal{U} \stackrel{\Delta}{=} \{\mathbf{U} \in \mathbb{R}^{n \times r}, \left\| \mathbf{U}_{:,k} \right\|_2 \leq 1, 1 \leq \kappa(\mathbf{U}) \leq \alpha\}$ with a constant α. The first constraint $\left\| \mathbf{U}_{:,k} \right\|_2 \leq 1$ is not restrictive as it is considered to bound the scale of basis vectors in $\mathbf{U}$ and hence prevents the arbitrarily very large values of $\mathbf{U}$. While the low condition number of the subspace $\kappa(\mathbf{U})$ is to prevent the ill-conditioned computation.

(A-3): Coefficients $\mathbf{w}$ are constrained to the set $\mathcal{W} = \{\mathbf{w} \in \mathbb{R}^r, \omega_1 \leq |w(i)| \leq \omega_2, i = 1, 2, \ldots, r\}$ with two constants ω_1 and ω_2, $0 \leq \omega_1 < \omega_2$. Since the data $\mathbf{x}$ and subspace $\mathbf{U}$ are assumed to be bounded, it is natural that the subspace weight vector $\mathbf{w}$ is bounded too.

(A-4): The subspace changes at two successive time instances is small, i.e., the largest principal angle between $\mathbf{U}_t$ and $\mathbf{U}_{t-1}$ is $0 \leq \theta_{\max} \ll \pi/2$, or the distance between the two subspaces, $\text{d}(\mathbf{U}_t, \mathbf{U}_{t-1}) = \sin(\theta_{\max})$, satisfies $0 \leq \text{d}(\mathbf{U}_t, \mathbf{U}_{t-1}) \ll 1$.

3.3   Proposed PETRELS-ADMM Algorithm

In this section, we present a novel algorithm, namely PETRELS-ADMM, for RST to handle missing data in the presence of outliers. The main idea is to minimize the empirical cost function g_t in (3.6) by updating outliers $\mathbf{s}_t$, weight vector $\mathbf{w}_t$ and subspace $\mathbf{U}_t$ alternatively.

Under the assumption (A-2) that the underlying subspace $\mathbf{U}$ changes slowly, we can detect outliers in $\mathbf{s}_t$ by projecting the new observation $\mathbf{x}_t$ into the space spanned by the formerly estimated subspace $\mathbf{U}_{t-1}$ in the previous phase. Specifically, we solve the following minimization problem:

$$\{\mathbf{s}_t, \mathbf{w}_t\} \overset{\Delta}{=} \underset{\mathbf{s}, \mathbf{w}}{\text{argmin}} \ \tilde{\ell}(\mathbf{U}_{t-1}, \mathbf{P}_t, \mathbf{x}_t, \mathbf{w}, \mathbf{s}) \tag{3.7}$$

with

$$\tilde{\ell}(\mathbf{U}_{t-1}, \mathbf{P}_t, \mathbf{x}_t, \mathbf{w}, \mathbf{s}) = \left\| \mathbf{P}_t (\mathbf{U}_{t-1}\mathbf{w} + \mathbf{s} - \mathbf{x}_t) \right\|_2^2 + \rho \left\| \mathbf{s} \right\|_1 . \tag{3.8}$$

In the second phase, the subspace $\mathbf{U}_t$ can be estimated by minimizing the sum of squared residuals:

$$\mathbf{U}_t = \underset{\mathbf{U}}{\text{argmin}} \ \frac{1}{t} \sum_{i=1}^{t} \beta^{t-i} \frac{\text{tr}(\widetilde{\mathbf{P}}_i)}{n} \left\| \widetilde{\mathbf{P}}_i (\mathbf{U}\mathbf{w}_i - \mathbf{x}_i) \right\|_2^2 + \frac{\alpha}{2t} \|\mathbf{U}\|_{2,\infty}^2, \tag{3.9}$$

where the regularization $\dfrac{\alpha}{2t} \|\mathbf{U}\|_{2,\infty}^2$ is to bound the scale of vectors in $\mathbf{U}$ while the outliers $\mathbf{s}_t$ has been disregarded and the new observation $\widetilde{\mathbf{P}}_i$ are determined by the following rule:

$$\begin{cases} \widetilde{\mathbf{P}}_i(k, k) = \mathbf{P}_i(k, k), & \text{if } \mathbf{s}_i(k) = 0, \\ \widetilde{\mathbf{P}}_i(k, k) = 0, & \text{otherwise,} \end{cases} \tag{3.10}$$

which we aim to skip the corrupted entries of $\mathbf{x}_i$.

Our algorithm first applies the ADMM framework in [114], which has been widely used in previous works for solving (3.7), and then propose a modification of PETRELS [73] to handle (3.9). In the outlier rejection stage, we emphasize here that we propose to focus on augmenting $\mathbf{s}$ (as shown in (3.12)) to further annihilate outlier effect, unlike GRASTA and ROSETA which focus on augmenting the residual error only.[3] Meanwhile, we modify the subspace update step in PETRELS by adding an adaptive step size $\eta_t \in (0, 1]$ at each time

---

[3]In GRASTA [50] and ROSETA [55], both the authors aimed to detect outliers $\mathbf{s}$ by solving the augmented Lagrangian of (3.7) as follows

$$\mathcal{L}(\mathbf{s}, \mathbf{y}, \mathbf{w}) = \|\mathbf{s}\|_1 + \mathbf{y}^\top \left(\mathbf{P}_t (\mathbf{U}_{t-1}\mathbf{w} + \mathbf{s} - \mathbf{x}_t) \right) + \frac{\rho}{2} \left\| \mathbf{P}_t (\mathbf{U}_{t-1}\mathbf{w} + \mathbf{s} - \mathbf{x}_t) \right\|_2^2 .$$

INPUT: Observed signals $\{\mathbf{x}_i\}_{i=1}^t$, $\mathbf{x}_i \in \mathbb{R}^{n \times 1}$, masks $\{\mathbf{P}_i\}_{i=1}^t$, $\mathbf{P}_i \in \mathbb{R}^{n \times n}$, rank r.

MAIN PROGRAM:

for $i = 1, 2, \ldots, t$

  // *Estimate outliers* $\mathbf{s}_i$ *and coefficient* $\mathbf{w}_i$ *using Algorithm 2:*

$$\{\mathbf{s}_i, \mathbf{w}_i\} = \underset{\mathbf{s},\mathbf{w}}{\operatorname{argmin}} \left\| \mathbf{P}_i (\mathbf{U}_{i-1} \mathbf{w} + \mathbf{s} - \mathbf{x}_i) \right\|_2^2 + \rho \left\| \mathbf{s} \right\|_1 .$$

  // *Update the new mask* $\widetilde{\mathbf{P}}_i$:

$$\begin{cases} \widetilde{\mathbf{P}}_i(k,k) = \mathbf{P}_i(k,k), & \text{if } \mathbf{s}_i(k) = 0, \\ \widetilde{\mathbf{P}}_i(k,k) = 0, & \text{otherwise.} \end{cases}$$

  // *Estimate subspace* $\mathbf{U}_i$ *using Algorithm 3:*

$$\mathbf{U}_i = \underset{\mathbf{U}}{\operatorname{argmin}} \left[\frac{1}{i} \sum_{j=1}^i \beta^{i-j} \frac{\operatorname{tr}(\widetilde{\mathbf{P}}_j)}{n} \left\| \widetilde{\mathbf{P}}_j(\mathbf{x}_j - \mathbf{U}\mathbf{w}) \right\|_2^2 + \frac{\alpha}{2i} \|\mathbf{U}\|_{2,\infty}^2 \right].$$

end for

OUTPUT: $\mathbf{U}_t \in \mathbb{R}^{n \times r}$

Algorithm 1: PETRELS-ADMM

instant t, instead of a constant one as in the original version. The modification can be interpreted as an approximation of Newton method. The proposed method is summarized in Algorithm 1.

3.3.1 Online ADMM for Outlier Detection

We show in the following how to solve (3.7) step-by-step:

Update $\mathbf{s}_t$

To estimate outlier $\mathbf{s}_t$ given $\mathbf{w}$, we exploit the fact that (3.7) can be cast into the ADMM form as follows:

$$\min_{\mathbf{u},\mathbf{s}} \quad h(\mathbf{u}) + q(\mathbf{s}) \quad \text{subject to } \mathbf{u} - \mathbf{s} = \mathbf{0}, \tag{3.11}$$

where $\mathbf{u}$ is the additional decision variable, $h(\mathbf{u}) = \frac{1}{2} \|\mathbf{P}_t (\mathbf{U}_{t-1}\mathbf{w} + \mathbf{u} - \mathbf{x}_t)\|_2^2$ and $q(\mathbf{s}) = \rho \|\mathbf{s}\|_1$. The corresponding augmented Lagrangian with the dual variable vector $\boldsymbol{\beta}$ is thus given by

$$\mathcal{L}(\mathbf{s}, \mathbf{u}, \boldsymbol{\beta}) = q(\mathbf{s}) + h(\mathbf{u}) + \boldsymbol{\beta}^\top (\mathbf{u} - \mathbf{s}) + \frac{\rho_1}{2} \|\mathbf{u} - \mathbf{s}\|_2^2, \tag{3.12}$$

INPUT: Observed signal $\mathbf{x}_t \in \mathbb{R}^{n \times 1}$, observation mask $\mathbf{P}_t \in \mathbb{R}^{n \times n}$, the previous estimate $\mathbf{U}_{t-1} \in \mathbb{R}^{n \times r}$, maximum iteration K, penalty parameters ρ_1, ρ_2, absolute and relative tolerances ϵ_{abs} and ϵ_{rel}.

INITIALIZATION:

- Choose $\{\mathbf{u}^0, \mathbf{s}^0, \mathbf{w}^0, \mathbf{z}^0, \mathbf{e}^0\}$ randomly.

- $\{\mathbf{r}^0, \mathbf{e}^0\} \leftarrow \mathbf{0}^n$

MAIN PROGRAM:

for $k = 0, 1, \dots, K$ **Cost**

// *Update* $\mathbf{w}$

$\quad \mathbf{w}^{k+1} = (\mathbf{P}_t\mathbf{U}_{i-1})^{\#}\mathbf{P}_t(\mathbf{x}_t - \mathbf{s}^k + \mathbf{e}^k)$ $2\Omega_t r^2 + \Omega_t r$

$\quad \mathbf{z}^{k+1} = \mathbf{P}_t(\mathbf{U}_{t-1}\mathbf{w}^{k+1} + \mathbf{s}^k - \mathbf{x}_t)$ $\Omega_t r$

$\quad \mathbf{e}^{k+1} = \frac{\rho_2}{1+\rho_2}\mathbf{z}^{k+1} + \frac{1}{1+\rho_2}S_{1+\frac{1}{\rho_2}}(\mathbf{z}^{k+1})$ Ω_t

// *Update* $\mathbf{s}$

$\quad \mathbf{u}^{k+1} = \frac{1}{1+\rho_1}\left(\mathbf{P}_t(\mathbf{x}_t - \mathbf{U}_{t-1}\mathbf{w}^{k+1}) - \rho_1(\mathbf{s}^k - \mathbf{r}^k)\right)$ $\Omega_t r$

$\quad \mathbf{s}^{k+1} = S_{\rho/\rho_1}(\mathbf{u}^{k+1} + \mathbf{r}^k)$ Ω_t

$\quad \mathbf{r}^{k+1} = \mathbf{r}^k + \mathbf{u}^{k+1} - \mathbf{s}^{k+1}$ Ω_t

// *Stopping criteria*

$\quad$ **if** $\left\|\mathbf{s}^{k+1} - \mathbf{s}^k\right\|_2 < \sqrt{n}\epsilon_{\text{abs}} + \epsilon_{\text{rel}}\left\|\rho_1\mathbf{r}^{k+1}\right\|_2$ break; Ω_t

$\quad$ **end if**

end for

OUTPUT: $\mathbf{s}, \mathbf{w}$

Algorithm 2: Outlier Detection

where $\rho_1 > 0$ is the regularization parameter[4]. Let $\mathbf{r} = \boldsymbol{\beta}/\rho_1$ be the scaled dual variable, we can rewrite the Lagrangian (3.12) as follows:

$$\mathcal{L}(\mathbf{s}, \mathbf{u}, \mathbf{r}) = q(\mathbf{s}) + h(\mathbf{u}) + \rho_1\mathbf{r}^\top(\mathbf{u} - \mathbf{s}) + \frac{\rho_1}{2}\|\mathbf{u} - \mathbf{s}\|_2^2. \tag{3.13}$$

The optimization of (3.13) is achieved iteratively where we have the following update rule using the scaled dual variable at the k-th iteration,

$$\mathbf{u}^{k+1} = \underset{\mathbf{u}}{\text{argmin}}\left(h(\mathbf{u}) + \rho_1(\mathbf{r}^k)^\top(\mathbf{u} - \mathbf{s}^k) + \frac{\rho_1}{2}\left\|\mathbf{u} - \mathbf{s}^k\right\|_2^2\right), \tag{3.14}$$

$$\mathbf{s}^{k+1} = \underset{\mathbf{s}}{\text{argmin}}\left(q(\mathbf{s}) - \rho_1(\mathbf{r}^k)^\top\mathbf{s} + \frac{\rho_1}{2}\left\|\mathbf{u}^{k+1} - \mathbf{s}\right\|_2^2\right), \tag{3.15}$$

$$\mathbf{r}^{k+1} = \mathbf{r}^k + \mathbf{u}^{k+1} - \mathbf{s}^{k+1}. \tag{3.16}$$

---

[4]It is referred to as the penalty parameter. Although the convergence rate of the proposed algorithm depends on a specific chosen value, our convergence analysis indicates that the ADMM solver can converge for any positive fixed penalty parameters. However, varying penalty parameters can give superior convergence in practice [114–117].

In particular, we first exploit that the minimization (3.14) can be formulated as a convex quadratic form:

$$
\begin{aligned}
\mathbf{u}^{k+1} &= \underset{\mathbf{u}}{\operatorname{argmin}} \left(\frac{1+\rho_1}{2} \|\mathbf{u}\|_2^2 - \left[\mathbf{P}_t(\mathbf{x}_t - \mathbf{U}_{t-1}\mathbf{w}) - \rho_1(\mathbf{s}^k - \mathbf{r}^k) \right]^\top \mathbf{u} \right) \\
&= \frac{1}{1+\rho_1} \left(\mathbf{P}_t(\mathbf{x}_t - \mathbf{U}_{t-1}\mathbf{w}) - \rho_1(\mathbf{s}^k - \mathbf{r}^k) \right).
\end{aligned}
\tag{3.17}
$$

While (3.15) is a standard proximal minimization with the ℓ_1-norm [118] as

$$
\begin{aligned}
\mathbf{s}^{k+1} &= \underset{\mathbf{s}}{\operatorname{argmin}} \left(\rho \|\mathbf{s}\|_1 + \frac{\rho_1}{2} \left\| \mathbf{s} - \left(\mathbf{u}^{k+1} + \mathbf{r}^k \right) \right\|_2^2 \right) \\
&= S_{\rho/\rho_1} \left(\mathbf{u}^{k+1} + \mathbf{r}^k \right),
\end{aligned}
\tag{3.18}
$$

where $S_a(x)$ is a thresholding operator applied element-wise and defined as

$$
S_a(x) = \begin{cases} 0, & \text{if } |x| \le a, \\ x - a, & \text{if } x > a, \\ x + a, & \text{if } x < -a, \end{cases}
\tag{3.19}
$$

which is a proximity operator of the ℓ_1-norm. Finally, a simple update rule for the scaled dual variable $\mathbf{r}$ can be given by the dual ascent, as

$$
\mathbf{r}^{k+1} = \mathbf{r}^k + \gamma^k \nabla_{\mathcal{L}}(\mathbf{r}^k),
\tag{3.20}
$$

where the gradient $\nabla_{\mathcal{L}}(\mathbf{r}^k)$ is computed by $\nabla_{\mathcal{L}}(\mathbf{r}^k) = \rho_1(\mathbf{u}^{k+1} - \mathbf{s}^{k+1})$ and $\gamma^k > 0$ is the step size controlling the convergence rate. For ADMM methods, the regularization parameter is often used as the the step size for updating dual variables [114]. Due to the scaled version $\mathbf{r}$ of the dual variable $\boldsymbol{\beta}$, the step size γ^k is here set to be $\gamma^k = 1/\rho_1$ at the k-th iteration.

Update $\mathbf{w}_t$

To estimate $\mathbf{w}_t$ given $\mathbf{s}$, (3.7) can be recast into the following ADMM form:

$$
\begin{aligned}
& \underset{\mathbf{w} \in \mathcal{W}, \mathbf{e} \in \mathbb{R}^{n \times 1}}{\min} \frac{1}{2} \left\| \mathbf{P}_t(\mathbf{U}_{t-1}\mathbf{w} + \mathbf{s} - \mathbf{x}_t) \right\|_2^2 + y(\mathbf{e}), \\
& \text{subject to} \quad \mathbf{P}_t(\mathbf{U}_{t-1}\mathbf{w} + \mathbf{s} - \mathbf{x}_t) = \mathbf{e},
\end{aligned}
\tag{3.21}
$$

where $y(\mathbf{e})$ is a convex regularizer function for the noise $\mathbf{e}$, (e.g. $y(\mathbf{e}) = \frac{\sigma}{2}\|\mathbf{e}\|_2^2$, with σ^{-1} can be chosen as the signal to noise ratio, SNR). The minimization (3.21) is equal to the following optimization:

$$
\underset{\mathbf{w} \in \mathcal{W}, \mathbf{e} \in \mathbb{R}^{n \times 1}}{\min} \|\mathbf{e}\|_2^2 \quad \text{subject to} \quad \mathbf{P}_t\left(\mathbf{U}_{t-1}\mathbf{w} + \mathbf{s} - \mathbf{x}_t\right) = \mathbf{e}.
\tag{3.22}
$$

However, the noise $\mathbf{e}$ is still affected by outliers because $\mathbf{s}$ may not be completely rejected in each iteration. Therefore, (3.22) can be cast further into the ADMM form such that it can lie between least squares (LS) and least absolute deviations to reduce the impact of outliers. The Huber fitting can bring transition between the quadratic and absolute terms of $\mathcal{L}_{\mathbf{w},\mathbf{e}}(\mathbf{w}, \mathbf{e})^5$, as

$$\mathcal{L}_{\mathbf{w},\mathbf{e}}(\mathbf{w}, \mathbf{e}) = f^{\text{Hub}}(\mathbf{e}) + \frac{\rho_2}{2}\left\|\mathbf{P}_t(\mathbf{U}_{t-1}\mathbf{w} + \mathbf{s} - \mathbf{x}_t) - \mathbf{e}\right\|_2^2, \qquad (3.23)$$

where $\rho_2 > 0$ is the penalty parameter whose characteristics are similar to that of ρ_1 and the Huber function is given by [114]

$$f^{\text{Hub}}(x) = \begin{cases} x^2/2, & |x| \le 1, \\ |x| - 1/2, & |x| > 1. \end{cases} \qquad (3.24)$$

As a result, $\mathbf{e}$-updates for estimating $\mathbf{w}$ involves the proximity operator of the Huber function, that is,

$$\mathbf{e}^{k+1} = \frac{\rho_2}{1 + \rho_2}\mathbf{P}_t(\mathbf{U}_{t-1}\mathbf{w}^{k+1} + \mathbf{s} - \mathbf{x}_t) + \frac{1}{1 + \rho_2}S_{1+\frac{1}{\rho_2}}\left(\mathbf{P}_t(\mathbf{U}_{t-1}\mathbf{w}^{k+1} + \mathbf{s} - \mathbf{x}_t)\right).$$
$$(3.25)$$

Hence, at the $(k + 1)$-th iteration, $\mathbf{w}^{k+1}$ can be updated using the following closed-form solution of the convex quadratic function:

$$\mathbf{w}^{k+1} = (\mathbf{P}_t\mathbf{U}_{t-1})^{\#}\mathbf{P}_t(\mathbf{x}_t - \mathbf{s} + \mathbf{e}^k). \qquad (3.26)$$

To sum up, the rule for updating $\mathbf{w}_t$ can be given by

$$\mathbf{w}^{k+1} = (\mathbf{P}_t\mathbf{U}_{t-1})^{\#}\mathbf{P}_t(\mathbf{x}_t - \mathbf{s} + \mathbf{e}^k), \qquad (3.27)$$

$$\mathbf{z}^{k+1} = \mathbf{P}_t(\mathbf{U}_{t-1}\mathbf{w}^{k+1} + \mathbf{s} - \mathbf{x}_t), \qquad (3.28)$$

$$\mathbf{e}^{k+1} = \frac{\rho_2}{1 + \rho_2}\mathbf{z}^{k+1} + \frac{1}{1 + \rho_2}S_{1+\frac{1}{\rho_2}}(\mathbf{z}^{k+1}). \qquad (3.29)$$

We note that, by using the Huber fitting operator, our algorithm is better in reducing the impact of outliers than GRASTA and ROSETA which use ℓ_2-norm regularization.

The procedure is stopped when the number of iterations reaches the maximum or the accuracy tolerance for the primal residual and dual norm has been met, i.e.,

$$\left\|\mathbf{s}^{k+1} - \mathbf{s}^k\right\|_2 < \sqrt{n}\epsilon_{\text{abs}} + \epsilon_{\text{rel}}\left\|\rho_1\mathbf{r}^{k+1}\right\|_2, \qquad (3.30)$$

---

[5]Due to the natural ℓ_2-ball behavior of the noise (i.e., normal distributed vector) and the sparsity of some unremoved parts of outliers, Huber fitting can be a reasonable choice. The Huber function consists of square and linear terms, so it is less sensitive to variables which have a strong effect on the function ℓ_2-norm, but also does not encourage the sparsity like ℓ_1-norm.

INPUT: Observed signals $\{\mathbf{x}_i\}_{i=1}^t$, observation mask $\widetilde{\mathbf{P}}_t$, the previous estimate $\mathbf{U}_{t-1}$, forgetting factor β, regularization parameter α, the step size η, ξ_t the previous matrix $\mathbf{H}_{t-1}$.

MAIN PROGRAM:	Cost
$x_t = \dfrac{\|\widetilde{\mathbf{P}}_t\mathbf{x}_t - \widetilde{\mathbf{P}}_t\mathbf{U}_{t-1}\mathbf{w}_t\|_2}{\|\mathbf{w}_t\|_2}$	$\Omega_t r$
$\eta_t = \dfrac{x_t}{\sqrt{x_t^2 + 1}}$	$O(1)$
if $\eta_t > \eta$ **then** $\eta_t = 1$ **end if**	$O(1)$
for $m = 1$ to n **do**	
$\quad \mathbf{R}_t^m = \beta\mathbf{R}_{t-1}^m + \widetilde{\mathbf{P}}_t(m,m)\mathbf{w}_t\mathbf{w}_t^\top$	r^2
$\quad \mathbf{H}_t^m = \mathbf{R}_t^m + \frac{\alpha}{2}\mathbf{I}$	r
$\quad \mathbf{a}_t = (\mathbf{H}_t^m)^{-1}\mathbf{w}_t$	$O(r^2)$
$\quad \mathbf{u}_t^m = \mathbf{u}_{t-1}^m + \eta_t\xi_t\widetilde{\mathbf{P}}_t(m,m)(\mathbf{x}_t^{\mathrm{re}}(m) - \mathbf{w}_t^\top\mathbf{u}_{t-1}^m)\mathbf{a}_t$	r
end for	

OUTPUT: $\mathbf{U}_t \in \mathbb{R}^{n \times r}$

Algorithm 3: Improved PETRELS for updating $\mathbf{U}_t$

where $\epsilon_{\mathrm{abs}} > 0$ and $\epsilon_{\mathrm{rel}} > 0$ are predefined tolerances for absolute and relative part respectively. A reasonable range for the absolute tolerance may be $[10^{-6}, 10^{-3}]$, while $[10^{-4}, 10^{-2}]$ is good for the relative tolerance, see [114] for further details of the stopping criterion. The main steps of the outlier detection are summarized as Algorithm 2.

3.3.2 Improved PETRELS for Subspace Estimation

Having estimated $\mathbf{s}_t$, we optimize the following minimization

$$\mathbf{U}_t := \underset{\mathbf{U}}{\mathrm{argmin}} \left[\tilde{g}_t(\mathbf{U}) = \frac{1}{t}\sum_{i=1}^t \beta^{t-i}\frac{\mathrm{tr}(\widetilde{\mathbf{P}}_i)}{n}\|\widetilde{\mathbf{P}}_i(\mathbf{x}_i - \mathbf{U}\mathbf{w}_i)\|_2^2 + \frac{\alpha}{2t}\|\mathbf{U}\|_{2,\infty}^2 \right], \tag{3.31}$$

where the observation mask $\widetilde{\mathbf{P}}_i$ is computed by (3.10).

Thanks to the parallel scheme of PETRELS [73], the optimal solution of the problem (3.31) can be obtained by solving its subproblems at each row $\mathbf{u}^m$ of $\mathbf{U}$, $1 \leq m \leq n$:

$$\mathbf{u}_t^m = \underset{\mathbf{u}^m}{\mathrm{argmin}} \frac{1}{t}\sum_{i=1}^t \beta^{t-i}\xi_i\widetilde{\mathbf{P}}_i(m,m)\left(\mathbf{x}_i(m) - \mathbf{w}_i^\top\mathbf{u}^m\right)^2 + \frac{\alpha}{2t}\|\mathbf{u}^m\|_2^2, \tag{3.32}$$

where $\xi_i = \frac{\mathrm{tr}(\widetilde{\mathbf{P}}_i)}{n}$. In this way, we can speed up the subspace update by ignoring the $\mathbf{u}^m$ if the m-th entry of $\mathbf{x}_t$ is labeled as missing observation or outlier.

Thanks to Newton's method, we can update each row of $\mathbf{U}_t$ by the following rule:

$$
\mathbf{u}_t^m = \mathbf{u}_{t-1}^m - \left[\mathbf{H}_t(\mathbf{u}^m)\right]^{-1} \frac{\partial \tilde{g}_t(\mathbf{U})}{\partial \mathbf{u}^m}\bigg|_{\mathbf{u}^m = \mathbf{u}_{t-1}^m}, \tag{3.33}
$$

where the first derivative of $\tilde{g}_t$ is given by

$$
\frac{\partial \tilde{g}_t(\mathbf{U})}{\partial \mathbf{u}^m} = \frac{-2}{t} \sum_{i=1}^{t} \beta^{t-i} \xi_i \widetilde{\mathbf{P}}_i(m,m)\left(\mathbf{x}_i(m) - \mathbf{w}_i^\top \mathbf{u}^m\right)\mathbf{w}_i^\top + \frac{\alpha}{t}\mathbf{u}^m, \tag{3.34}
$$

and the second derivative of $\tilde{g}_t$, Hessian matrix, is given by

$$
\mathbf{H}_t(\mathbf{u}^m) = \frac{2}{t} \sum_{i=1}^{t} \beta^{t-i} \xi_i \widetilde{\mathbf{P}}_i(m,m)\mathbf{w}_i \mathbf{w}_i^\top + \frac{\alpha}{t}\mathbf{I}. \tag{3.35}
$$

Specifically, the partial derivative $\frac{\partial \tilde{g}_t(\mathbf{U})}{\partial \mathbf{u}^m}$ at $\mathbf{u}_{t-1}^m$ can be expressed by

$$
\begin{aligned}
\frac{\partial \tilde{g}_t(\mathbf{U})}{\partial \mathbf{u}^m}\bigg|_{\mathbf{u}^m = \mathbf{u}_{t-1}^m} &= \frac{\partial \tilde{g}_{t-1}(\mathbf{U})}{\partial \mathbf{u}^m}\bigg|_{\mathbf{u}^m = \mathbf{u}_{t-1}^m} \\
&+ \frac{\alpha}{t}\left(\mathbf{u}_{t-1}^m - \mathbf{u}_{t-2}^m\right) - \frac{2}{t}\xi_t \widetilde{\mathbf{P}}_t(m,m)\left(\mathbf{x}_t(m) - \mathbf{w}_t^\top \mathbf{u}_{t-1}^m\right)\mathbf{w}_t^\top.
\end{aligned} \tag{3.36}
$$

Since $\mathbf{u}_{t-1}^m = \mathrm{argmin} \frac{\partial \tilde{g}_{t-1}(\mathbf{U})}{\partial \mathbf{u}^m}$ and the parameter α/t is small, so $\frac{\partial \tilde{g}_{t-1}(\mathbf{U})}{\partial \mathbf{u}^m}\big|_{\mathbf{u}^m = \mathbf{u}_{t-1}^m}$ = $\mathbf{0}$ and then

$$
\frac{\partial \tilde{g}_t(\mathbf{U})}{\partial \mathbf{u}^m}\bigg|_{\mathbf{u}^m = \mathbf{u}_{t-1}^m} \approx \frac{-2}{t}\xi_t \widetilde{\mathbf{P}}_t(m,m)\left(\mathbf{x}_t(m) - \mathbf{w}_t^\top \mathbf{u}_{t-1}^m\right)\mathbf{w}_t^\top. \tag{3.37}
$$

Let us denote $\mathbf{R}_t^m = \sum_{i=1}^{t} \beta_i^{t-i} \xi_i \widetilde{\mathbf{P}}_t(m,m)\mathbf{w}_i \mathbf{w}_i^\top$, the Hessian matrix can be rewritten by

$$
\mathbf{H}_t^m \overset{\Delta}{=} \mathcal{H}\tilde{f}_t(\mathbf{u}_{t-1}^m) = \frac{2}{t}\left(\mathbf{R}_t^m + \frac{\alpha}{2}\mathbf{I}\right). \tag{3.38}
$$

Therefore, a relaxed approximation of the recursive update (3.33) is given by

$$
\mathbf{u}_t^m \approx \mathbf{u}_{t-1}^m + \eta_t \xi_t \widetilde{\mathbf{P}}_t(m,m)\left(\mathbf{x}_t(m) - \mathbf{w}_t^\top \mathbf{u}_{t-1}^m\right)\mathbf{a}_t^\top, \tag{3.39}
$$
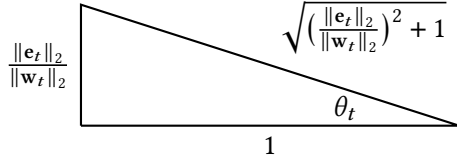
Figure 3.1: Adaptive step size η_t.

where $\mathbf{H}_t^m = \mathbf{R}_t^m + \frac{\alpha}{2}\mathbf{I}^6$, $\mathbf{a}_t = (\mathbf{H}_t^m)^{-1}\mathbf{w}_t$ and η_t denotes the adaptive step size $\eta_t \in [0, 1]$ at each time instant t, instead of a constant as in the original PETRELS [73]. We here determine the adaptive step size η_t as follows

$$\eta_t = \frac{x_t}{\sqrt{x_t^2 + 1}} \text{ with } x_t = \frac{\|\mathbf{e}_t\|_2}{\|\mathbf{w}_t\|_2}, \qquad (3.40)$$

where the residual error $\mathbf{e}_t$ is computed by $\mathbf{e}_t = \widetilde{\mathbf{P}}_t\mathbf{x}_t - \widetilde{\mathbf{P}}_t\mathbf{U}_{t-1}\mathbf{w}_t$. Note that, the adaptive step size η_t can be expressed by $\eta_t = \sin(\theta_t)$, see Fig. 3.1. The smaller angle θ_t is, the closer to the true subspace we are, the smaller step size is needed. The update is summarized in Algorithm 3.

3.3.3   Computational Complexity Analysis

The number of floating-point operations (flop) is used to measure the computational complexity of the proposed PETRELS-ADMM. At the k-th iteration in the outlier detection phase, our method requires $O(\Omega r^2)$ flops where Ω is average number of observed entries at each time instant ($\Omega \leq n$). It is practically stated that the ADMM solver can converge within a few tens of iterations [114] (also see Fig. 3.3). Therefore, the removal of outliers costs the averaged $O(\Omega r^2)$. The complexity of the subspace estimation phase is also $O(\Omega r^2)$ as the original PETRELS [73]. The overall computational complexity of PETRELS-ADMM is of order $O(\Omega r^2)$ flops.

3.4   Performance Analysis

In this section, we provide a convergence analysis for the proposed PETRELS-ADMM algorithm. Inspired by the results of convergence of empirical processes for online sparse coding in [120] and online robust PCA in [121, 122], we derive a theoretical approach to analyze the convergence of values of the

---

[6]$\mathbf{H}_t^m \in \mathbb{R}^{r\times r}$ is a matrix of rank-one updates, so its inverse matrix can be efficiently computed recursively, thanks to Sherman–Morrison formula [119]. Also, the small regularization parameter $\alpha > 0$ can help the recursive update having a better numerical stability. The computational complexity is of order $O(r^2)$.

objective function $\{f_t(\mathbf{U}_t)\}_{t=1}^\infty$ as well as the solutions $\{\mathbf{U}_t\}_{t=1}^\infty$ generated by PETRELS-ADMM.

Given assumptions defined in Section 3.2.2, our main theoretical result can be stated by the following theorem:

> **Theorem 2 (Convergence of PETRELS-ADMM)** *In the stationary context, let $\{\mathbf{U}_t\}_{t=1}^\infty$ be the sequence of solutions generated by PETRELS-ADMM, then the sequence converges to a stationary point of the expected loss function $f(\mathbf{U})$ when $t \to \infty$.*

Proof Sketch. Our proof can be divided into three main stages as follows: We first prove that the solutions $\{\mathbf{U}_t, \mathbf{s}_t\}_{t\geq 1}$ generated by the PETRELS-ADMM algorithm are optimal w.r.t. the cost function in (3.6). We then prove that a nonnegative sequence $\{g_t(\mathbf{U}_t)\}_{t=1}^\infty$ converges almost surely where $\{\mathbf{U}_t\}_{t=1}^\infty$ is the sequence of optimal solutions generated by the PETRELS-ADMM algorithm. After that, we prove that the surrogate $\{g_t(\mathbf{U}_t)\}_{t=1}^\infty$ converges almost surely to the empirical loss function $\{f_t(\mathbf{U}_t)\}_{t=1}^\infty$ as well as the true loss function, i.e., $g_t(\mathbf{U}_t) \overset{a.s.}{\to} f_t(\mathbf{U}_t) \overset{a.s.}{\to} f(\mathbf{U}_t)$, thanks to the central limit theorem.

Due to space limitation, we here present key results and report their proof sketch. The details of their proofs are provided in our appendix.

> **Lemma 1 (Convergence of Algorithm 2)** *At each time t, let $\{\mathbf{s}^k, \mathbf{u}^k, \mathbf{r}^k, \mathbf{w}^k, \mathbf{e}^k\}_{k=1}^\infty$ be a sequence generated by Algorithm 2 for outlier detection, there always exists a set of positive numbers $\{c_u, c_s, c_r, c_w, c_e\}$ such that, at each iteration, the minimizers satisfy*
>
> $$\mathcal{L}\big(\mathbf{s}^{k+1}, \mathbf{u}^{k+1}, \mathbf{r}^{k+1}, \mathbf{w}^{k+1}, \mathbf{e}^{k+1}\big) \leq \mathcal{L}\big(\mathbf{s}^k, \mathbf{u}^k, \mathbf{r}^k, \mathbf{w}^k, \mathbf{e}^k\big) - c_u \big\|\mathbf{u}^k - \mathbf{u}^{k+1}\big\|_2^2$$
> $$- c_s \big\|\mathbf{s}^k - \mathbf{s}^{k+1}\big\|_2^2 - c_r \big\|\mathbf{r}^k - \mathbf{r}^{k+1}\big\|_2^2$$
> $$- c_w \big\|\mathbf{w}^k - \mathbf{w}^{k+1}\big\|_2^2 - c_e \big\|\mathbf{e}^k - \mathbf{e}^{k+1}\big\|_2^2, \tag{3.41}$$
>
> *where the Lagrangian $\mathcal{L}(\mathbf{s}, \mathbf{u}, \mathbf{r}, \mathbf{w}, \mathbf{e})$ for updating these variables is a combination of two functions (3.13) and (3.23), as*
>
> $$\mathcal{L}\big(\mathbf{s}, \mathbf{u}, \mathbf{r}, \mathbf{w}, \mathbf{e}\big) = q(\mathbf{s}) + h(\mathbf{u}) + \rho_1 \mathbf{r}^\top(\mathbf{u} - \mathbf{s}) + \frac{\rho_1}{2}\|\mathbf{u} - \mathbf{s}\|_2^2$$
> $$+ f^{Hub}(\mathbf{e}) + \frac{\rho_2}{2}\big\|\mathbf{P}_t\big(\mathbf{U}_{t-1}\mathbf{w} + \mathbf{s} - \mathbf{x}_t\big) - \mathbf{e}\big\|_2^2. \tag{3.42}$$
>
> *The asymptotic variation of $\mathbf{s}^k$ (i.e., outliers) is then given by*
>
> $$\lim_{k \to \infty} \big\|\mathbf{s}^{k+1} - \mathbf{s}^k\big\|_2^2 = 0. \tag{3.43}$$

Proof Sketch. We state the following proposition, which is in the same line as in previous convergence analysis of ADMM algorithms [123, 124], used to prove the first part of lemma 1.

Proposition 1 *Let* $\{\mathbf{s}^k, \mathbf{u}^k, \mathbf{r}^k, \mathbf{w}^k, \mathbf{e}^k\}_{k=1}^{\infty}$ *be a sequence generated by Algorithm 2 and denote* $\mathbf{q}^k$ *be one of these variables, the minimizer* $\mathbf{q}^{k+1}$ *of* (3.13) *satisfies*

$$\mathcal{L}(\mathbf{q}^{k+1}, .) \leq \mathcal{L}(\mathbf{q}^k, .) - c_q \|\mathbf{q}^k - \mathbf{q}^{k+1}\|_2^2, \qquad (3.44)$$

where c_q *is a positive number.*

As a result, the cluster $\{\mathbf{s}^k, \mathbf{u}^k, \mathbf{r}^k, \mathbf{w}^k, \mathbf{e}^k\}$ converges to stationary point of $\mathcal{L}(\mathbf{s}, \mathbf{u}, \mathbf{r}, \mathbf{w}, \mathbf{e})$ when $k \to \infty$ and it also implies that the sequence $\{\mathbf{s}_k\}_{k=0}^{\infty}$ is convergent, i.e.,

$$\lim_{k \to \infty} \left\| \mathbf{s}^{k+1} - \mathbf{s}^k \right\|_2^2 = 0. \qquad (3.45)$$

Proposition 2 (Convexity of the surrogate functions $g_t(\mathbf{U})$)
Given assumptions in Section 3.2.2, the surrogate function $g_t(\mathbf{U})$ *defined in Eq.* (3.6) *is not only strongly convex, but also Lipschitz function, i.e., there always exists two positive numbers* m_1 *and* m_2 *such that*

$$m_1 \|\mathbf{U}_{t+1} - \mathbf{U}_t\|_F^2 \leq \left|g_t(\mathbf{U}_{t+1}) - g_t(\mathbf{U}_t)\right|, \qquad (3.46)$$

$$m_2 \|\mathbf{U}_{t+1} - \mathbf{U}_t\|_F \geq \left|g_t(\mathbf{U}_{t+1}) - g_t(\mathbf{U}_t)\right|. \qquad (3.47)$$

Proof Sketch. To prove that $g_t(\mathbf{U})$ is strongly convex, we state the following facts: $g_t(\mathbf{U})$ is continuous and differentiable; its second derivative is a positive semi-definite matrix (i.e., $\nabla_{\mathbf{U}}^2 g_t(\mathbf{U}) \succeq m\mathbf{I}$); and the domain of $g_t(\mathbf{U})$ is convex. In order to satisfy the Lipschitz condition, we show that the first derivative of $g_t(\mathbf{U})$ is bounded.

Lemma 2 (Convergence of Algorithm 3) *Given an outlier vector* $\mathbf{s}_t$ *generated by Algorithm 2 at each time instant t, Algorithm 3 can provide a local optimal solution* $\mathbf{U}_t$ *for minimizing* $g_t(\mathbf{U})$. *Moreover, the asymptotic variation of estimated subspaces* $\{\mathbf{U}_t\}_{t \geq 1}$ *is given by*

$$\|\mathbf{U}_t - \mathbf{U}_{t+1}\|_F \xrightarrow{a.s.} O\left(\frac{1}{t}\right). \qquad (3.48)$$

Proof Sketch. To establish the convergence, we exploit the fact that our modification can be seen as an approximate of the Newton method,

$$\mathbf{U}_t \cong \mathbf{U}_{t-1} - \eta_t \left[\mathcal{H} \tilde{f}_t(\mathbf{U}_{t-1}) \right]^{-1} \nabla \tilde{g}_t(\mathbf{U}_{t-1}), \tag{3.49}$$

where $\mathcal{H}\tilde{f}_t(\mathbf{U}_{t-1})$ and $\nabla \tilde{g}_t(\mathbf{U}_{t-1})$ are the Hessian matrix and gradient of the function $\tilde{g}_t(\mathbf{U})$ at $\mathbf{U}_{t-1}$, as shown in Section 3.3.2. It implies that the estimated $\mathbf{U}_t$ converges to the stationary point of $g_t(\mathbf{U})$.

Furthermore, since $g_t(\mathbf{U})$ is strongly convex and Lipschitz function w.r.t the variable $\mathbf{U}$ as shown in Proposition 2, we have the following inequality

$$m_1 \|\mathbf{U}_{t+1} - \mathbf{U}_t\|_F^2 \leq \left| g_t(\mathbf{U}_{t+1}) - g_t(\mathbf{U}_t) \right| \leq m_2 \|\mathbf{U}_{t+1} - \mathbf{U}_t\|_F \tag{3.50}$$

$$\Leftrightarrow \|\mathbf{U}_t - \mathbf{U}_{t+1}\|_F \leq \frac{m_2}{m_1} = O\left(\frac{1}{t}\right). \tag{3.51}$$

Note that the positive number $m_2 = O(1/t)$ is already given in the proof of Proposition 2 in the supplemental material, while m_1 is a constant.

Lemma 3 (Convergence of the surrogate function $g_t(\mathbf{U})$)
Without discounting past observations, let $\{\mathbf{U}_t\}_{t=1}^{\infty}$ be a sequence of solutions generated by Algorithm 1 at each time instant t, the sequence $\{g_t(\mathbf{U}_t)\}_{t=1}^{\infty}$ converges almost surely, i.e.,

$$\sum_{t=1}^{\infty} \left| \mathbb{E}\left[g_{t+1}(\mathbf{U}_{t+1}) - g_t(\mathbf{U}_t) \middle| \mathcal{F}_t \right] \right| < +\infty \quad a.s., \tag{3.52}$$

where $\{\mathcal{F}_t\}_{t>0}$ is the filtration of the past estimations at time instant t.

Proof Sketch. Let us define the indicator function δ_t as follows

$$\delta_t \triangleq \begin{cases} 1 & \text{if } \mathbb{E}\left[g_{t+1}(\mathbf{U}_{t+1}) - g_t(\mathbf{U}_t) \middle| \mathcal{F}_t \right] > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.53}$$

According to the quasi-martingale convergence theorem [125, Section 4.4], in order to show the convergence of the nonnegative stochastic process $\{g_t(\mathbf{U}_t)\}_{t=1}^{\infty}$, we will prove

$$\sum_{t=0}^{\infty} \mathbb{E}\left[\delta_t \mathbb{E}\left[g_{t+1}(\mathbf{U}_{t+1}) - g_t(\mathbf{U}_t) \middle| \mathcal{F}_t \right] \right] < \infty. \tag{3.54}$$

In particular, we first indicate the following inequality:

$$g_{t+1}(\mathbf{U}_{t+1}) - g_t(\mathbf{U}_t) \leq \frac{\ell(\mathbf{U}_t, \mathbf{P}_{t+1}, \mathbf{x}_{t+1}) - f_t(\mathbf{U}_t)}{t+1}. \tag{3.55}$$

Since $\mathbb{E}\big[\ell(\mathbf{U}_t, \mathbf{P}_{t+1}, \mathbf{x}_t)\big] = f(\mathbf{U}_t)$, we have a nice property:

$$\mathbb{E}\Big[g_{t+1}(\mathbf{U}_{t+1}) - g_t(\mathbf{U}_t)\big|\mathcal{F}_t\Big] \leq \frac{\mathbb{E}\big[\ell(\mathbf{U}_t, \mathbf{P}_{t+1}, \mathbf{x}_{t+1}) - f_t(\mathbf{U}_t)\big|\mathcal{F}_t\big]}{t+1}$$
$$= \frac{f(\mathbf{U}_t) - f_t(\mathbf{U}_t)}{t+1}. \tag{3.56}$$

We then have

$$\mathbb{E}\Big[\delta_t \mathbb{E}\big[g_{t+1}(\mathbf{U}_{t+1}) - g_t(\mathbf{U}_t)\big|\mathcal{F}_t\big]\Big] \leq \mathbb{E}\Big[\sqrt{t}\big(f(\mathbf{U}_t) - f_t(\mathbf{U}_t)\big)\Big]\frac{1}{\sqrt{t}(t+1)}. \tag{3.57}$$

Under the given assumptions, we exploit the fact that the set of measurable functions $\{\ell(\mathbf{U}_i, \mathbf{P}, \mathbf{x})\}_{i\geq 1}$ defined in (3.2) is $\mathbb{P}$-Donsker. Therefore, the centered and scaled version of the empirical function $f_t(\mathbf{U}_t)$ satisfies the following proposition:

$$\mathbb{E}\Big[\sqrt{t}\big(f(\mathbf{U}_t) - f_t(\mathbf{U}_t)\big)\Big] = \mathcal{O}(1), \tag{3.58}$$

thanks to Donsker theorem [126, Sec 19.2]. Furthermore, we also indicate that the sum $\sum_{t=1}^{\infty} 1/(\sqrt{t}(t+1))$ converges. The two facts result in

$$\sum_{t=0}^{\infty} \mathbb{E}\Big[\delta\mathbb{E}\big[g_{t+1}(\mathbf{U}_{t+1}) - g_t(\mathbf{U}_t)\big|\mathcal{F}_t\big]\Big] < \infty. \tag{3.59}$$

Since $g_t(\mathbf{U}_t) > 0$, we can conclude that $\{g_t(\mathbf{U}_t)\}_{t>0}$ is quasi-martingale and converges almost surely.

Lemma 4 (Convergence of the empirical loss function $f_t(\mathbf{U})$)
The empirical loss functions $f_t(\mathbf{U}_t)$ and its surrogate $g_t(\mathbf{U}_t)$ converge to the same limit, i.e.,

$$g_t(\mathbf{U}_t) \xrightarrow{a.s.} f_t(\mathbf{U}_t). \tag{3.60}$$

Proof Sketch. We begin the proof with providing the following inequality:

$$\frac{g_t(\mathbf{U}_t) - f_t(\mathbf{U}_t)}{t+1} \leq \underbrace{u_t - u_{t+1}}_{\text{(S-1)}} + \underbrace{\frac{\ell(\mathbf{U}_t, \mathbf{P}_{t+1}, \mathbf{x}_{t+1}) - f_t(\mathbf{U}_t)}{t+1}}_{\text{(S-2)}}, \tag{3.61}$$

where $u_t \triangleq g_t(\mathbf{U}_t)$. We then prove that the two sequences (S-1)-(S-2) converge almost surely. As a result, the sequence $\big\{(g_t(\mathbf{U}_t) - f_t(\mathbf{U}_t))\frac{1}{t+1}\big\}$ also convergence almost surely, i.e.,

$$\sum_{t=0}^{\infty} \big(g_t(\mathbf{U}_t) - f_t(\mathbf{U}_t)\big)\frac{1}{t+1} < \infty. \tag{3.62}$$

In parallel, we exploit that the real sequence $\{\frac{1}{t+1}\}_{t\geq 0}$ diverges, i.e., $\sum_{t=1}^{\infty}\frac{1}{t+1} = \infty$. It implies that $g_t(\mathbf{U}_t) - f_t(\mathbf{U}_t)$ converges.

Corollary 1 *The expected loss function $\{f(\mathbf{U}_t)\}_{t=1}^{\infty}$ converges almost surely when $t \to \infty$.*

Proof. Since $f_t(\mathbf{U}_t) \overset{a.s.}{\to} f(\mathbf{U}_t)$ and $g_t(\mathbf{U}_t) \overset{a.s.}{\to} f_t(\mathbf{U}_t)$, then $g_t(\mathbf{U}_t) \overset{a.s.}{\to} f(\mathbf{U}_t)$. Since $g_t(\mathbf{U}_t)$ converges almost surely, $f(\mathbf{U}_t)$ also converges almost surely when $t \to \infty$.

Corollary 2 *When $t \to \infty$, let $\mathbf{U}_t = \arg\min_{\mathbf{U}\in\mathbb{R}^{n\times r}} g_t(\mathbf{U})$, we have*

$$f_t(\mathbf{U}_t) \leq f_t(\mathbf{U}) + \frac{L}{2}\|\mathbf{U} - \mathbf{U}_t\|_F^2, \forall\, \mathbf{U} \in \mathbb{R}^{n\times r}, \tag{3.63}$$

where L is a positive constant. In other words, $\mathbf{U}_t$ is the minimum point of $f(\mathbf{U})$.

Proof. Let us denote the error function $e_t(\mathbf{U}) = g_t(\mathbf{U}) - f_t(\mathbf{U})$. Due to $g_t(\mathbf{U}_t) \overset{a.s.}{\to} f_t(\mathbf{U}_t)$ when $t \to \infty$, we have $\nabla e_t(\mathbf{U}_t) = \mathbf{0}$ and hence the following inequality

$$\|\nabla e_t(\mathbf{U})\| \leq \frac{L}{2}\|\mathbf{U} - \mathbf{U}_t\|_F. \tag{3.64}$$

It is therefore that

$$\frac{|e_t(\mathbf{U}) - e_t(\mathbf{U}_t)|}{\|\mathbf{U} - \mathbf{U}_t\|_F} \leq \frac{L}{2}\|\mathbf{U} - \mathbf{U}_t\|_F, \tag{3.65}$$

thanks to the mean value theorem. In other word, we have $|e_t(\mathbf{U})| \leq \frac{L}{2}\|\mathbf{U} - \mathbf{U}_t\|_F^2$ because of $e_t(\mathbf{U}_t) \overset{a.s.}{\to} 0$.

In addition, for all $\mathbf{U} \in \mathbb{R}^{n\times r}$, we always have $f_t(\mathbf{U}_t) \leq g_t(\mathbf{U})$. Therefore, we can conclude the corollary as follows

$$f_t(\mathbf{U}_t) \leq g_t(\mathbf{U}_t) = f_t(\mathbf{U}) + e_t(\mathbf{U}) \leq f_t(\mathbf{U}) + \frac{L}{2}\|\mathbf{U} - \mathbf{U}_t\|_F^2. \tag{3.66}$$

It ends the proof.

3.5   Experiments

In this section, we evaluate the performance of the proposed algorithm by comparing it to the state-of-the-art in three scenarios relative to: robust subspace tracking, robust matrix completion and video background-foreground

separation respectively. In particular, extensive experiments on simulated data are conducted to demonstrate the convergence and robustness of our PETRELS-ADMM algorithm for subspace tracking and matrix completion. While four real video sequences are used to illustrate the effectiveness of PETRELS-ADMM for background-foreground separation.

3.5.1 Robust Subspace Tracking

In the following experiments, data $\mathbf{x}_t$ at each time t is generated randomly using the standard signal model as in (3.1)

$$\mathbf{x}_t = \mathbf{P}_t(\mathbf{U}\boldsymbol{\omega}_t + \mathbf{n}_t + \mathbf{s}_t), \tag{3.67}$$

where $\mathbf{U} \in \mathbb{R}^{n \times r}$ denotes a mixing matrix, $\boldsymbol{\omega}_t$ is a random vector living on $\mathbb{R}^r$ space (i.e., $\boldsymbol{\ell}_t = \mathbf{U}\boldsymbol{\omega}_t$) and they are Gaussian i.i.d. of pdf $\mathcal{N}(0,1)$; $\mathbf{n}_t$ represents the white Gaussian noise $\mathcal{N}(0,\sigma^2)$, with SNR $= -10\log_{10}(\sigma^2)$ is the signal-to-noise ratio to control the impact of noise on algorithm performance; and $\mathbf{s}_t$ is uniform i.i.d. over $[0, \text{fac-outlier}]$ given the magnitude fac-outlier of outliers that aim to create a space for outliers. Indices of missing entries and outliers are generated randomly using the Bernoulli model with the probability ω_{missing} and ω_{outlier} respectively. The two probabilities represent the density of missing entries and outliers in the data.

In order to evaluate the subspace estimation accuracy, we use the subspace estimation performance (SEP) [62] metric

$$\text{SEP} = \frac{1}{L}\sum_{i=1}^{L}\frac{\text{tr}\big\{\mathbf{U}_{\text{es-i}}^{\#}(\mathbf{I} - \mathbf{U}_{\text{ex}}\mathbf{U}_{\text{ex}}^{\#})\mathbf{U}_{\text{es-i}}\big\}}{\text{tr}\big\{\mathbf{U}_{\text{es-i}}^{\#}(\mathbf{U}_{\text{ex}}\mathbf{U}_{\text{ex}}^{\#})\mathbf{U}_{\text{es-i}}\big\}}, \tag{3.68}$$

where L is the number of independent runs, $\mathbf{U}_{\text{ex}}$ and $\mathbf{U}_{\text{es-i}}$ are the true and the estimated subspaces at the i-th run respectively. Particularly, the denominator measures the sum of the squares of the cosines of the principal angles between $\mathbf{U}_{\text{es-i}}$ and $\mathbf{U}_{\text{ex}}$, while the numerator evaluates the similar sum but for the two subspaces $\mathbf{U}_{\text{es-i}}$ and the orthogonal complement $\mathbf{U}_{\text{ex}}^{\perp}$. Accordingly, the lower SEP is, the better the algorithm performance is.

The state-of-the-art RST algorithms for comparison are: GRASTA [50], ROSETA [55] and PETRELS-CFAR [62], ReProCS [63] and NORST [64]. In our experiments, their algorithm parameters are set by default as mentioned in the algorithms. In particular, we set a penalty parameter $\rho = 1.8$ and a constant step-size scale $C = 2$ in GRASTA. An adaptive step size of ROSETA is initialized at $\mu_0 = \frac{C}{1+\eta_0}$ with $C = 8$ and $\eta_0 = 99$, while two thresholds for controlling the step size are set at $\eta_{\text{low}} = 50$ and $\eta_{\text{high}} = 100$. PETRELS-CFAR includes a forgetting factor set at $\lambda = 0.999$, a window size $N_w = 150$ and a false alarm probability P_{fa} varied from $[0.1, 0.7]$ depended on the outlier

(a) SNR = 0 dB.

(b) SNR = 10 dB.
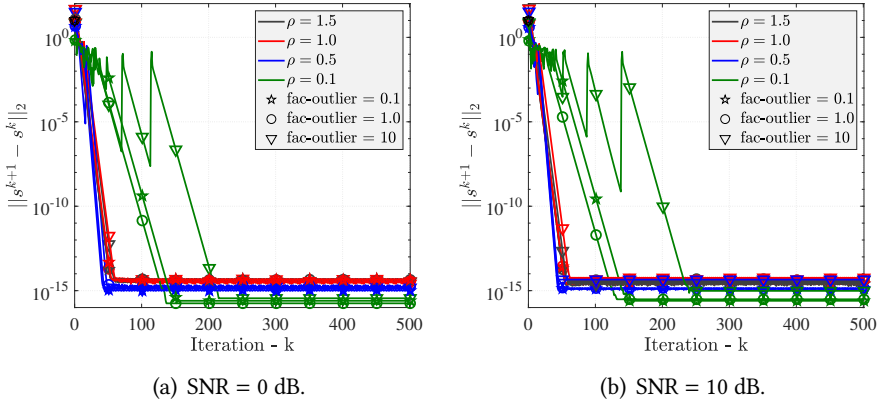
Figure 3.2: Convergence of PETRELS-ADMM in terms of the variation $\|\mathbf{s}^{k+1} - \mathbf{s}^k\|_2$: $n = 50, r = 2, 90\%$ **entries observed and outlier density** $\omega_{\text{outlier}} = 0.1$.

intensity. Both ReProCS and NORST require several predefined parameters, including $t_{\text{train}} = 200$ data samples, $\alpha = 60, K = 33$ and $\omega_{eval} = 7.8 \times 10^{-4}$. For our algorithm, we set the penalty parameters at 1.5, the regularization parameter $\alpha = 0.1$ and the step-size threshold $\eta = \sin(\pi/3)$, while the maximum number of iterations for outlier detection phase is fixed at $K = 50$. Matlab codes are available online[7]. The experimental results are averaged over 100 independent runs.

3.5.1.1 Convergence of PETRELS-ADMM

To demonstrate the convergence of our algorithm, we use a synthetic data whose number of row $n = 50$, rank $r = 2$, and 5000 vector samples with 90% entries observed on average. Specifically, the outlier density ω_{outlier} is varied from 0.05 to 0.4, while the outlier intensity is set at three values representing a low, medium and high level (i.e., fac-outlier = 0.1, 1 and 10 respectively). The penalty parameter ρ varies in the range $[0.1, 1.5]$. Also, two noise levels are considered, with SNR $\in \{0, 10\}$ dB. The results are shown as in Fig. 3.2 and Fig. 3.3.

Fig. 3.2 shows the convergence behavior of PETRELS-ADMM w.r.t the two variables: fac-outlier and the weight ρ. We can see that, the variation of $\{\mathbf{s}^k\}_{k \geq 1}$ always converges in all testing cases. When the penalty parameter

---

[7]GRASTA: https://sites.google.com/site/hejunzz/grasta
ROSETA: http://www.merl.com/research/license#ROSETA
ReProCS: https://github.com/praneethmurthy/ReProCS
Our code: https://github.com/thanhtbt/RST

(a) Outlier density $\omega_{\text{outlier}} = 0.05$. (b) Outlier density $\omega_{\text{outlier}} = 0.4$.
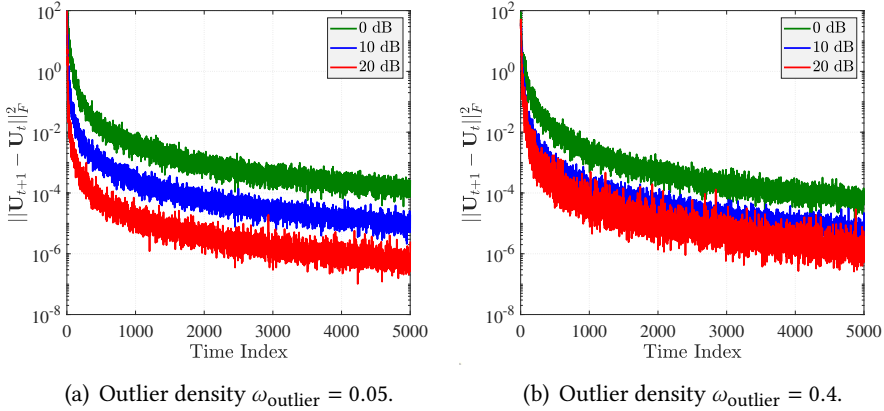
Figure 3.3: Convergence of PETRELS-ADMM in terms of the variation $\|\mathbf{U}_{t+1} - \mathbf{U}_t\|_F$**:** $n = 50, r = 2$, 90% **entries observed and outlier intensity fac-outlier = 10.**

$\rho \geq 0.5$, the convergence rate is fast, i.e. the variation $\left\|\mathbf{s}^{k+1} - \mathbf{s}^k\right\|_2$ can converge in 50 iterations in both low- and high-noise cases. The results are practical evidences of Lemma 1. Similarly, Fig. 3.3 shows that the convergence of the variations of the sequence $\{\mathbf{U}_t\}_{t \geq 0}$, generated by PETRELS- ADMM follows the theoretical behavior proved in Lemma 2, that is, $\|\mathbf{U}_t - \mathbf{U}_{t+1}\|_F \overset{a.s.}{\to} \mathcal{O}(1/t)$ almost surely.

3.5.1.2 Outlier Detection

Following the above experiment, we next assess the ability of PETRELS-ADMM for outlier detection against the noise level. The three statistical metrics including Sensitivity (SEN) and Specificity (SEP) and Accuracy (ACC) are used to evaluate its outlier detection performance [127]. Particularly, SEN measures the percentage of outliers detected correctly over the total outliers in the measurement data. SEP is similar to SEN, but for normal entries and ACC indicates how the estimator makes the correct detection. We use the same data above, but 20% of the observations are missing. The outlier density ω_{outlier} is set at 0.2, while two intensity levels are considered, with fac-outlier $\in \{1, 10\}$.

Fig. 3.4 illustrates the outlier detection performance of PETRELS-ADMM versus the noise level SNR. As can be seen that when we increase the value of SNR from -20 dB to 20 dB, the detection accuracy goes up first and then converges towards a constant level. At very low SNRs (i.e., < 0 dB), the proposed algorithm does not work well in which many normal entries are labeled as outliers, although the number of correctly detected outliers are high. When

(a) fac-outlier = 1.   (b) fac-outlier = 10.
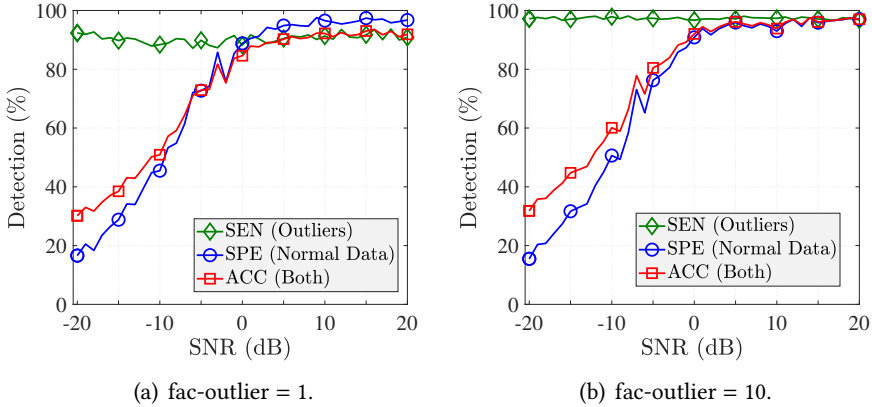
Figure 3.4: Outlier detection accuracy versus the noise level: $n = 50$, $r = 2$, 80% **entries observed and** 20% **outliers.**

SNR > 0 dB, PETRELS-ADMM achieves a competitive prediction accuracy with respect to all three evaluation metrics.

Fig. 3.5 provides more practical evidences to demonstrate the effectiveness of PETRELS-ADMM for the outlier detection. Particularly, the locations of outliers $\mathbf{s}_t$ are well detected even when the measurement data is corrupted by noise with a moderate SNR value (e.g. 10 dB). Also, amplitude of the outliers is recovered nearly correctly with a small relative error (RE = $\frac{\|\mathbf{s}_t - \tilde{\mathbf{s}}_t\|_2}{\|\mathbf{s}_t\|_2}$) in both cases (e.g. RE = 0.0616 at the 20 dB noise level). As a result, the corrupted signals are also well reconstructed, as illustrated in Fig. 3.5(b) and (d).

3.5.1.3   Robustness of PETRELS-ADMM

To investigate the robustness of PETRELS-ADMM, we vary the outlier intensity, density and missing density and then measure the SEP metric. Moreover, we also demonstrate the effectiveness of PETRELS-ADMM against noisy and time-varying environments.

Impact of outlier intensity on algorithm performance

We fix $n = 50$, $r = 2$, 90% entries observed, outlier density $\omega_{\text{outlier}} = 0.1$, SNR= 20 dB while varying fac-outlier in the range [0.1, 10]. We can see from Fig. 3.6 that PETRELS-ADMM always outperforms other state-of-the-art algorithms in all testing cases with different fac-outlier values. At low outlier intensity (i.e., fac-outlier ≤ 1), all algorithms yield good accuracy with fast convergences, though ROSETA and ReProCS obtain the higher SEP (i.e., $\approx 10^{-3}$) as compared to that of the four remaining algorithms. In partic-

(a) Outlier detection: SNR = 20 dB.

(b) Data recovery: SNR = 20 dB.

(c) Outlier detection: SNR = 10 dB.

(d) Data recovery: SNR = 10 dB.
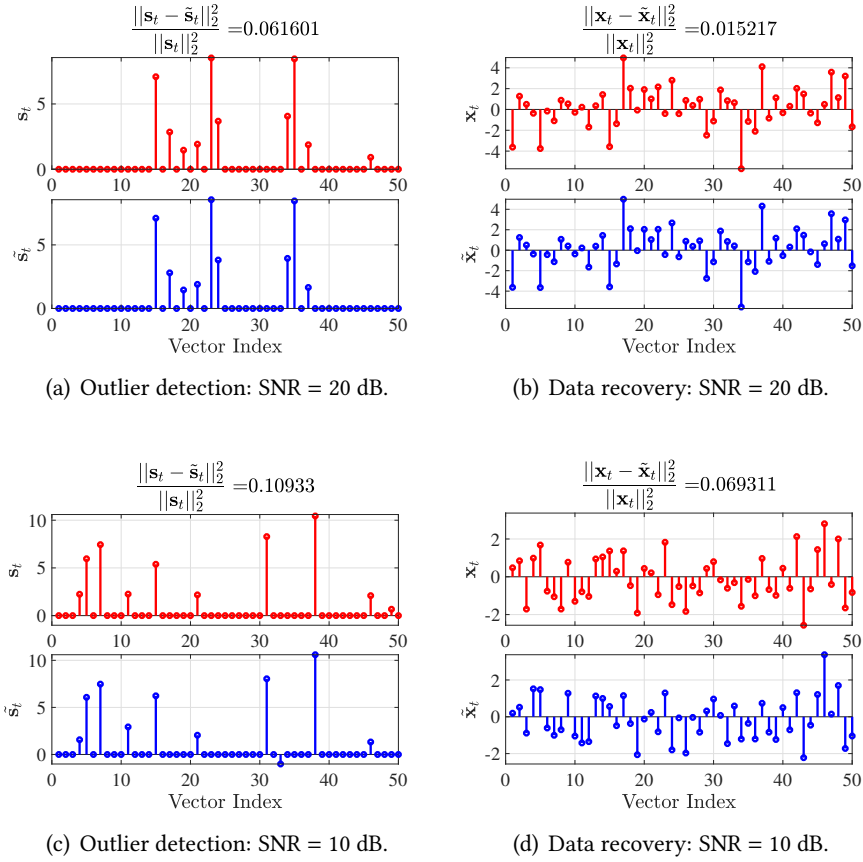
Figure 3.5: Outlier detection and data reconstruction: $n = 50$, $r = 2$, 90% **entries observed, outlier intensity** fac-outlier = 1, **and outlier density** $\omega_{\text{outlier}} = 0.1$.

ular, PETRELS-ADMM provides the best subspace estimation accuracy, i.e., SEP $\approx 10^{-5}$ in both cases (see Fig. 3.6(a)-(b)). At a high intensity level (e.g. fac-outlier = 5 or 10), PETRELS-ADMM again provides the best performance in terms of both convergence rate and accuracy. GRASTA performs similarly to ReProCS and slightly worse than PETRELS-CFAR (i.e., their SEP values are around 10^{-4}). While ROSETA and NORST fail to recover the underlying subspace in the presence of strong outliers. Note that, in all four experiments above, PETRELS-ADMM always obtains the best SEP value of around 10^{-5} and hence is robust to outlier intensity.

(a) fac-outlier = 0.1.

(b) fac-outlier = 1.

(c) fac-outlier = 5.

(d) fac-outlier = 10.
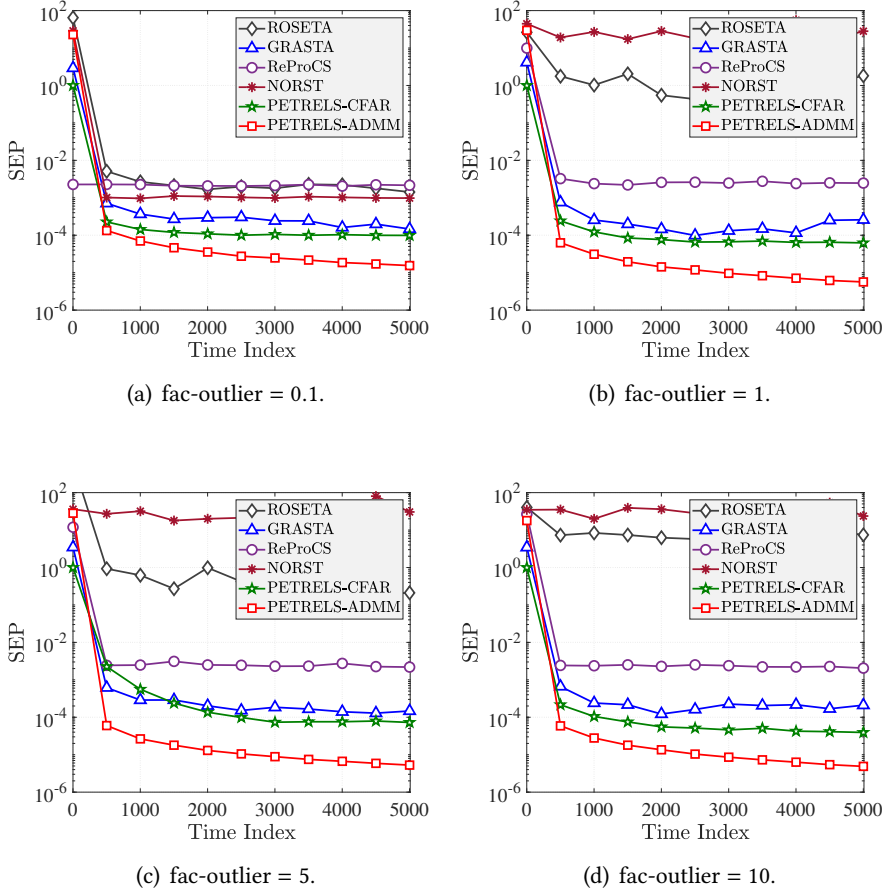
Figure 3.6: Impact of outlier intensity on algorithm performance: $n = 50$, $r = 2$, 90% **entries observed, outlier density** $\omega_{\text{outlier}} = 0.1$ **and SNR = 20 dB.**

Impact of outlier density on algorithm performance

We fix $n = 50$, $r = 2$, 90% entries observed, outlier intensity fac-outlier = 5, SNR = 20 dB while varying the outlier density ω_{outlier} in the range [0.05, 0.4]. The results are shown as in Fig. 3.7. PETRELS-ADMM outperforms the four remaining algorithms in this context. In particular, our algorithm performs very well even when the fraction of outliers is high (e.g. $\omega_{\text{outlier}} = 0.4$). By contrast, four algorithms including GRASTA, ROSETA, ReProCS and NORST may fail to track subspace in the case of a high outlier density (see Fig. 3.7(d)). The PETRELS-CFAR works well but has a lower convergence rate and accuracy in terms of SEP metric as compared to PETRELS-ADMM. When the measurement data is corrupted by a smaller number of outliers, PETRELS-ADMM
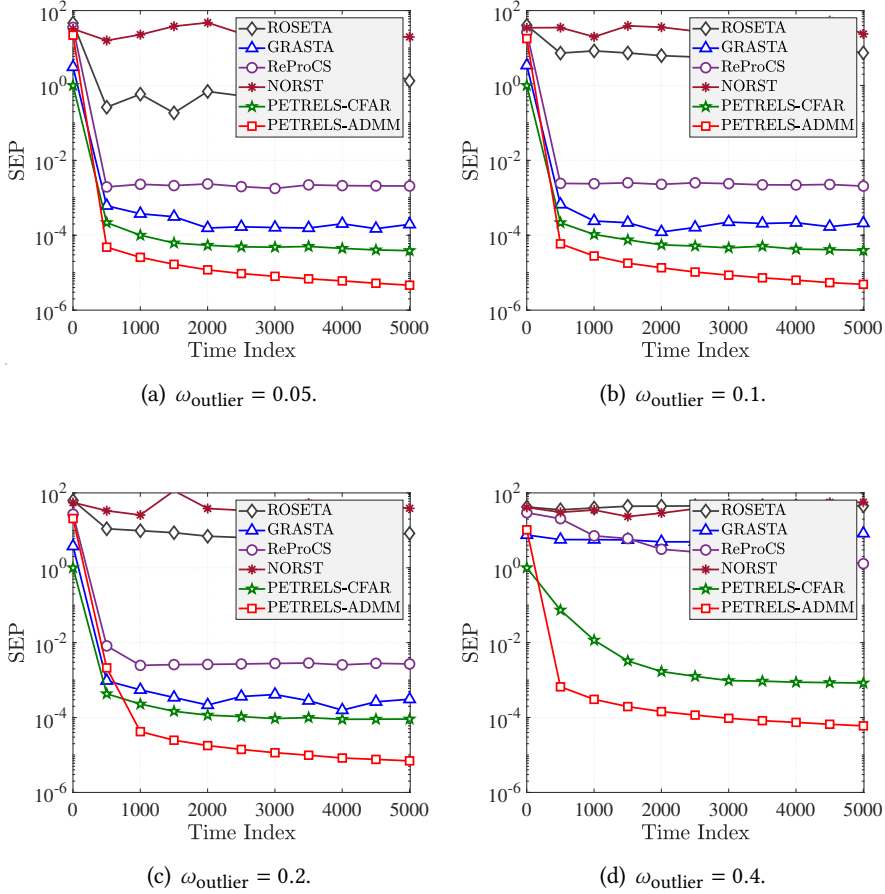
Figure 3.7: **Impact of outlier density on algorithm performance:** $n = 50, r = 2$, 90% **entries observed, outlier intensity** fac-outlier $= 10$ **and** SNR $= 20$ **dB.**

still provides better performance than the others, as shown in Fig. 3.7 (a)-(c).

Impact of the density of missing entries on algorithm performance

Following the above experiments, we change the number of missing entries in the measurement data by varying the probability $\omega_{missing}$ while fixing the other attributes. The results are reported in Fig. 7.13 and Fig. 3.9. In particular, the effect of $\omega_{missing}$ on algorithm performance is presented in Fig. 7.13. Similarly, PETRELS-ADMM yields the best performance in four cases of missing observations. Three algorithms including PETRELS-CFAR, GRASTA and Re-ProCS provide good performance but with slower convergence rate and accuracy, while ROSETA and NORST have failed again in this task due to the high
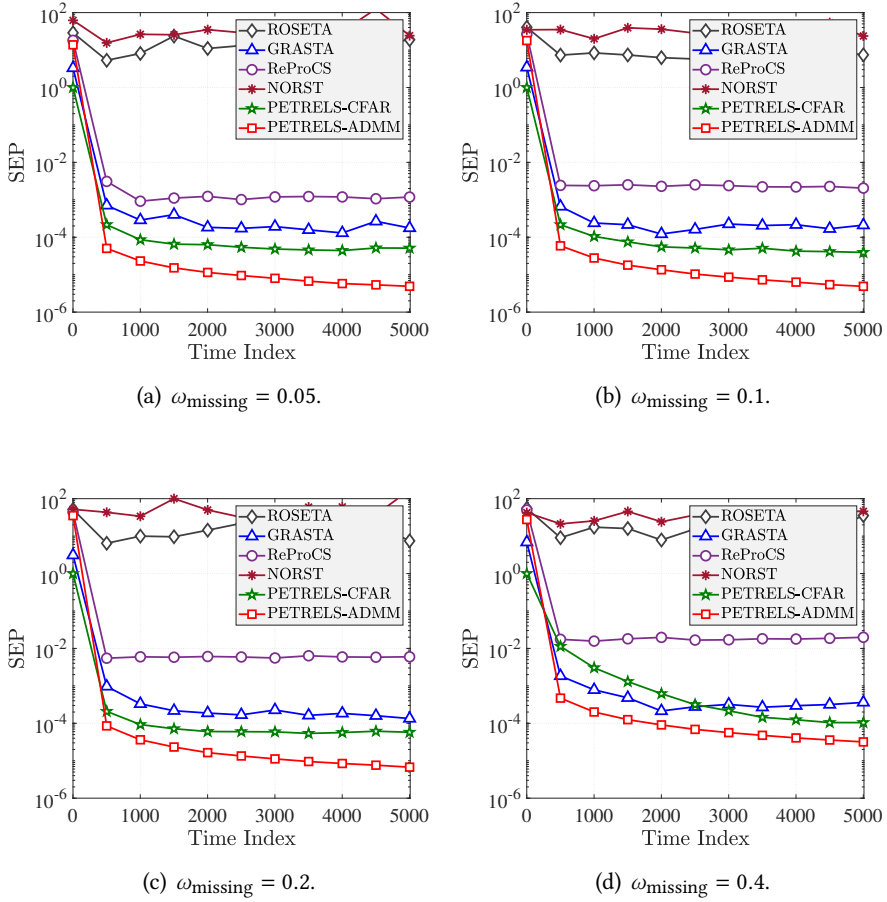
Figure 3.8: Impact of the density of missing entries on algorithm performance: $n = 50, r = 2$, **outlier density** $\omega_{\text{outlier}} = 0.1$, **outlier intensity** fac-outlier = 10 **and** SNR = 20 **dB.**

outlier intensity (i.e., fac-outlier = 10). As can be seen from Fig. 3.9(a)-(c) that the state-of-the-art algorithms only perform well when the number of corruptions is smaller than half the number of entries in the data measurement. While PETRELS-ADMM still obtains the reasonable subspace estimation performance in terms of SEP (i.e., $\approx 10^{-3}$) in the case of very high corruptions, see Fig. 3.9(d).
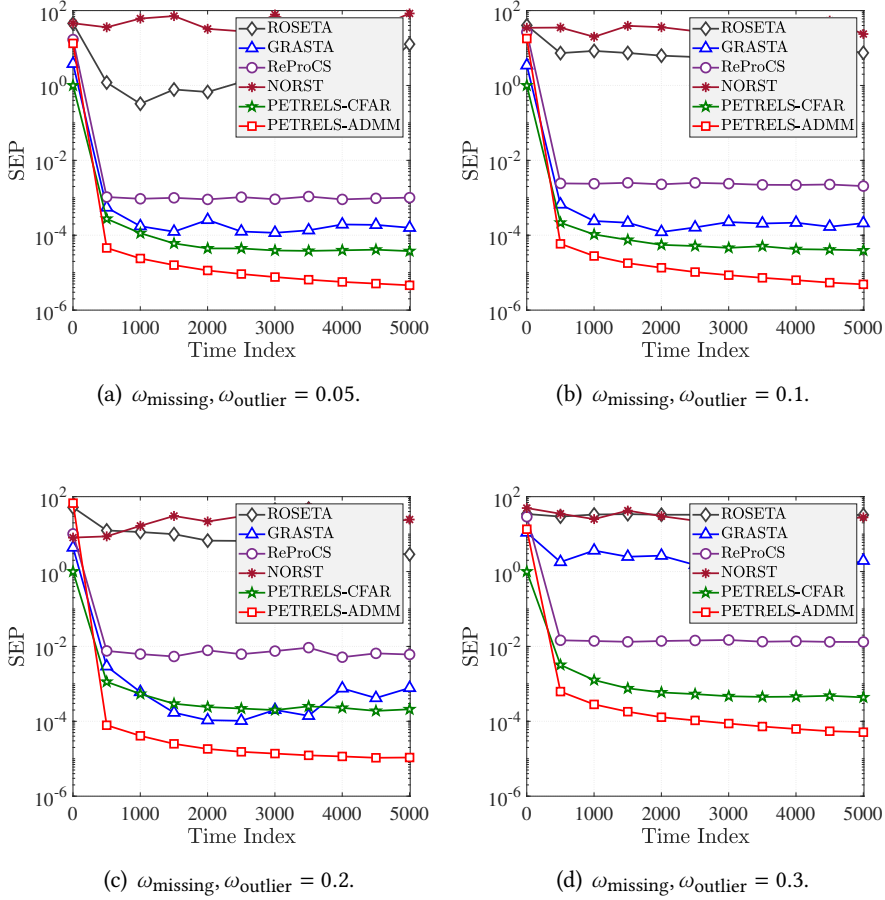
Figure 3.9: **Impact of the corruption fraction by missing data and outliers on algorithm performance:** $n = 50, r = 2$ **and** fac-outlier = 10 **and** SNR = 20 **dB.**

Noisy and Time-Varying Environments

We first investigate the effect of the noise on the performance of PETRELS-ADMM in comparison with the state-of-the-art algorithms. We vary the value of SNR in the range from 0 dB to 20 dB and assess their performance on the same data above. Experimental results are illustrated in Fig. 3.10. As can be seen that the convergence rate of PETRELS-ADMM is not affected by SNR, but only its estimation accuracy, as shown in Fig. 3.10(a). Specifically, when we decrease the value of SNR, the estimation error between the true subspace and the estimation increases gradually. At a high SNR level (e.g. 20 dB), previous experiments indicate that PETRELS-ADMM outperforms state-of-the-art

(a) PETRELS-ADMM.

(b) SNR = 5 dB.
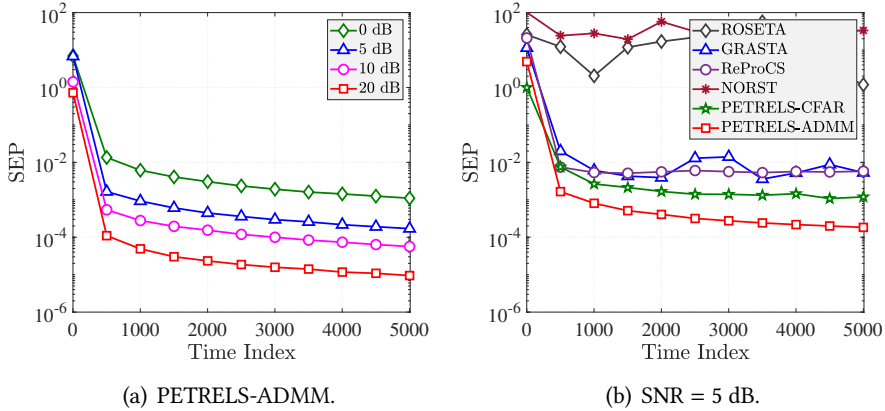
Figure 3.10: Impact of the additive noise on algorithm performance: $n = 50, r = 2$**,** 90% **entries observed and** 10% **outliers with intensity** fac-outlier = 10**.**

algorithms, see Fig. 3.6-3.9. At a low SNR level (e.g. 5 dB), PETRELS-ADMM yields the best estimation accuracy as well as convergence rate again, as illustrated in Fig. 3.10(b). Similar outstanding performance of PETRELS-ADMM were also observed at lower SNR levels of 10, 5 or 0 dB (please see Figs. 8-10 of the supplementary material).

The robustness of PETRELS-ADMM is next investigated against nonstationary and time-varying environments. Particularly, the true subspace $\mathbf{U}$ is supposed to be varying with time under the model $\mathbf{U}_t = (1 - \varepsilon)\mathbf{U}_{t-1} + \varepsilon\mathbf{N}_t$, where $\mathbf{N}_t \in \mathbb{R}^{n \times r}$ is a Gaussian noise matrix (zero-mean and unit-variance) and ε is to control the subspace change which is chosen among $\{10^{-1}, 10^{-2}, 10^{-3}\}$. We use the same signal model as in the previous tasks and 1000 vector samples. Also, we create an abrupt change at $t = 500$ to see how fast the proposed algorithm can converge. We measure the performance of PETRELS-ADMM at two noise levels (SNR = 5 and 10 dB) with different corruption fractions. Experimental results are illustrated in Fig. 7.12(a)-(d). In the same manner to the effect of the noise, the time-varying factor ε does not affect the convergence rate of PETRELS-ADMM, but only its subspace estimation. Fig. 7.12 shows that the estimation accuracy of the proposed algorithm will decrease if the time-varying factor ε increases. When the underlying subspace varies slowly (e.g. $\varepsilon \leq 10^{-2}$), the resulting values of SEP, which always converge towards an error floor, indicate that PETRELS-ADMM can be robust to slowly time-varying scenarios.
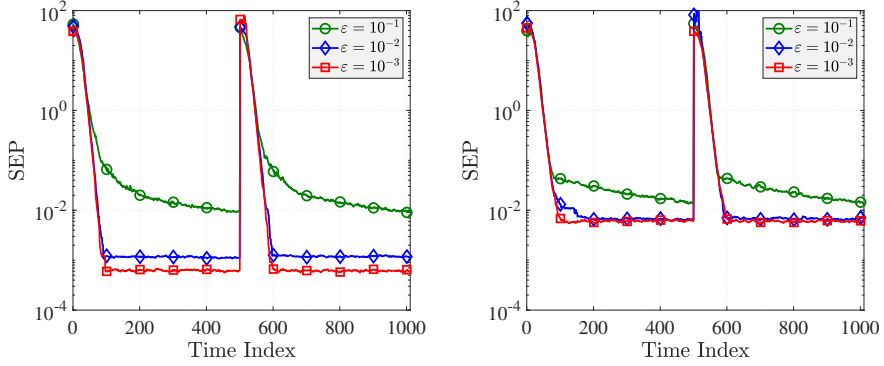
(a) SNR = 10 dB, ω_{missing} = 0.05 and ω_{outlier} = 0.05.

(b) SNR = 5 dB, ω_{missing} = 0.05 and ω_{outlier} = 0.05.
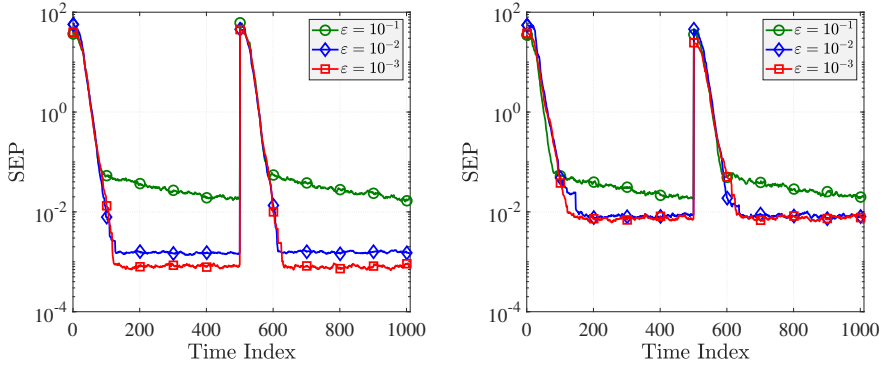
(c) SNR = 10 dB, ω_{missing} = 0.2 and ω_{outlier} = 0.2.

(d) SNR = 5 dB, ω_{missing} = 0.2 and ω_{outlier} = 0.2.

Figure 3.11: PETRELS-ADMM in time-varying scenarios.

3.5.2 Robust Matrix Completion

We compare here the robust matrix completion (RMC) performance using PETRELS-ADMM with GRASTA [50], LRGeomGC [128] and RPCA-GD [129].

The measurement data $\mathbf{X} = \mathbf{P} \circledast (\mathbf{UW}+\mathbf{S}+\mathbf{N})$ used for this task corresponds to the rank-2 matrices of size of 400×400, where the operator $\circledast$ denotes the Hadamard product. Particularly, we generated the mixing matrix $\mathbf{U} \in \mathbb{R}^{400 \times 2}$ and the coefficient matrix $\mathbf{W} \in \mathbb{R}^{2 \times 400}$ at random. Their entries were random variables that follow Gaussian distribution with zero mean and unit variance. The measurement data $\mathbf{X}$ was corrupted by a white Gaussian noise $\mathbf{N} \in \mathbb{R}^{400 \times 400}$ whose SNR is fixed at 40 dB. In the literature, the SNR value of around 40 dB is used for performance evaluation of completion algorithms
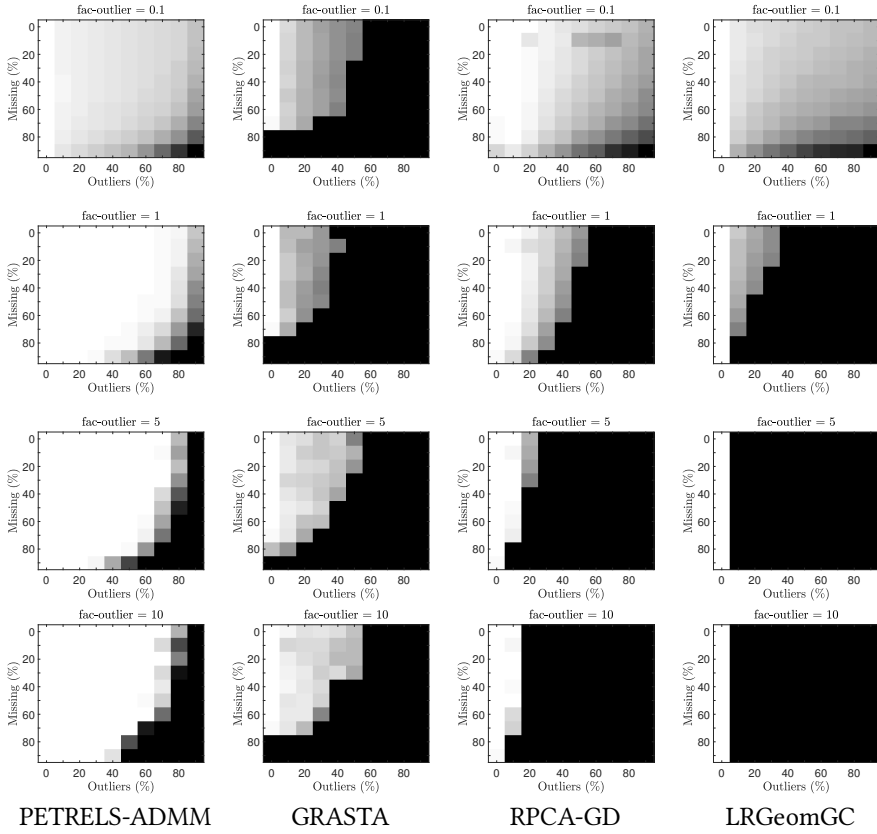
Figure 3.12: Effect of outlier intensity on robust matrix completion performance. White color denotes perfect recovery, black color denotes failure and gray colour is in between.

due to missing observations and/or outliers at low-noise conditions [130]. The data matrix was affected by different percentages of missing (**P**) and outliers (**S**) from 0% − 90%. The location and value of corrupted entries (including missing and outliers) were uniformly distributed.

Fig. 3.12 shows that the proposed algorithm of PETRELS-ADMM based RMC outperforms GRASTA, LRGeomGC and RPCA-GD. At low outlier intensity (i.e., fac-outlier = 0.1), PETRELS-ADMM based RMC, LRGeomGC and RCPA-GD provide excellent performance even when the data is corrupted by a very high corruption fraction. At high outlier intensity (i.e., fac-outlier ≥ 1), PETRELS-ADMM based RMC provides the best matrix reconstruction error performance, GRASTA still retain good performance, while RPCA-GD and LRGeomGC fail to recover corrupted entries.

3.5.3   Video Background/Foreground Separation

We further illustrate the effectiveness of the proposed PETRELS-ADMM algorithm in the application of RST for video background/foreground separation, and compare with GRASTA and PETRELS-CFAR. We use four real video sequences for this task, including `Hall`, `Lobby`, `Sidewalk` and `Highway` datasets. In particular, the two former datasets are from GRASTA's homepage[8], while the two latter datasets are from CD.net2012[9] [131]. The `Hall` dataset consists of 3584 frames of size 174×144 pixels, while the `Lobby` dataset has 1546 frames of size 144×176 pixels. The `Sidewalk` dataset includes 1200 frames of size 240×352 pixels. `Highway dataset` has 1700 frames of size 240×320 pixels. We can see from Fig. 3.13, PETRELS-ADMM is capable of detecting objects in video and provides competitive performance as compared to GRASTA and PETRELS-CFAR.

3.6   Conclusions

In this chapter, we have proposed an efficient algorithm, namely PETRELS-ADMM, for the robust subspace tracking problem to handle missing data in the presence of outliers. By converting the original RST problem to a surrogate one, which facilitates the tracking ability, we have derived an online implementation for outlier rejection with a low computational complexity and a fast convergence rate while still retaining a high subspace estimation performance. We have established a theoretical convergence which guarantees that the solutions generated by PETRELS-ADMM will converge to a stationary point asymptotically. The simulation results have suggested that our algorithm is more effective than the state-of-the-art algorithms for robust subspace tracking and robust matrix completion. The effectiveness of PETRELS-ADMM was also verified for the problem of video background-foreground separation.

3.7   Appendix

3.7.1   Proof of Lemma 1

Follow the line as in previous convergence analysis of ADMM algorithms [123, 124], we can derive the proof of Lemma 1 as follows

---

[8]https://sites.google.com/site/hejunzz/grasta
[9]http://jacarini.dinf.usherbrooke.ca/dataset2012

Video Frame   PETRELS-ADMM   PETRELS-CFAR   GRASTA
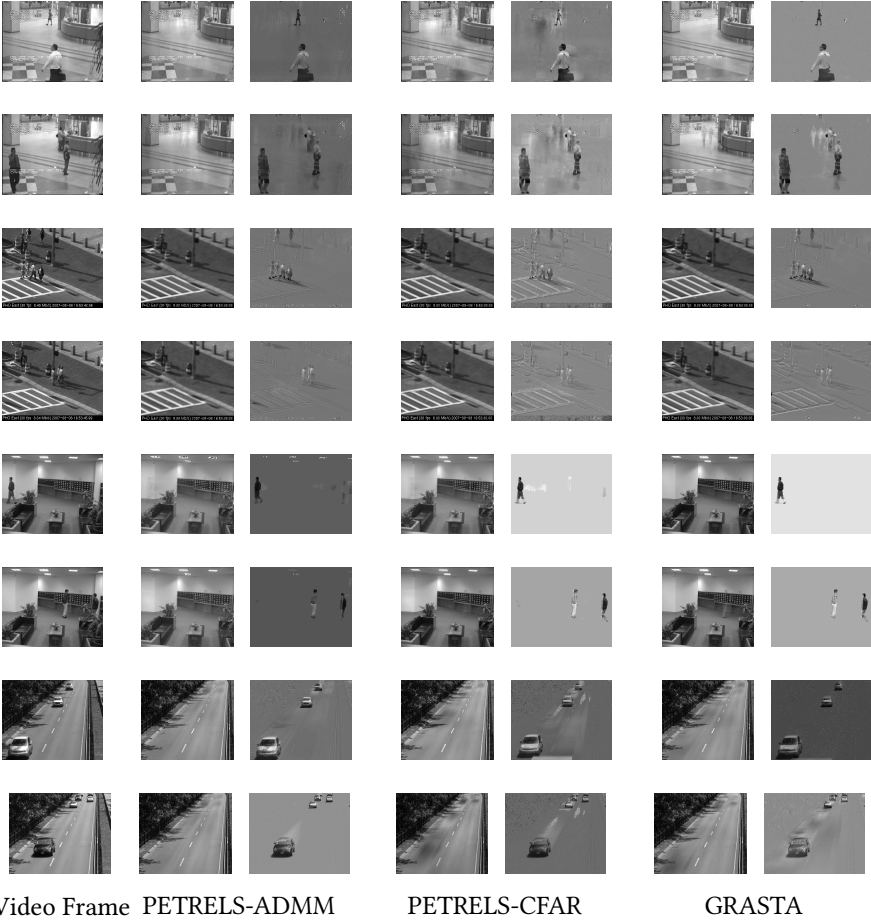
Figure 3.13: Qualitative illustration of video background-foreground separation application.

3.7.1.1   Proof of Proposition (P-1)

The minimizer $\mathbf{u}^{k+1}$ defined in (3.15) satisfies

$$\mathcal{L}\big(\mathbf{s}^k, \mathbf{u}^{k+1}, \mathbf{r}^k, \mathbf{w}^k, \mathbf{e}^k\big) \leq \mathcal{L}(\mathbf{s}^k, \mathbf{u}^k, \mathbf{r}^k, \mathbf{w}^k, \mathbf{e}^k) - c_u\big\|\mathbf{u}^k - \mathbf{u}^{k+1}\big\|_2^2. \qquad \text{(P-1)}$$

At the k-th iteration, the $\mathbf{u}$-update in fact minimizes the objective function in (3.14), as

$$\mathbf{u}^{k+1} = \underset{\mathbf{u}}{\operatorname{argmin}} \left[\mathcal{L}_{\mathbf{u},k}(\mathbf{u}, .) = \frac{1+\rho_1}{2} \|\mathbf{u}\|_2^2 - \left[\mathbf{P}_t\big(\mathbf{x}_t - \mathbf{U}_{t-1}\mathbf{w}\big) - \rho_1\big(\mathbf{s}^k - \mathbf{r}^k\big) \right]^\top \mathbf{u} \right].$$
$$\text{(3.69)}$$

The function $\mathcal{L}_{\mathbf{u},k}(\mathbf{u}, .)$ is strongly convex with a positive constant $(1 + \rho_1)$, i.e., the Hessian of $\mathcal{L}_{\mathbf{u},k}(\mathbf{u}, .)$ is given by $\nabla^2 \mathcal{L}_{\mathbf{u},k}(\mathbf{u}, .) = (1+\rho_1)\mathbf{I}$. Since $\mathbf{u}^{k+1} =$

argmin$_\mathbf{u}$ $\mathcal{L}_{\mathbf{u},k}(\mathbf{u},.)$, we have the fact $\mathcal{L}_{\mathbf{u},k}(\mathbf{u}^{k+1},.) \leq \mathcal{L}_{\mathbf{u},k}(\mathbf{u}^k,.)$. Therefore, we obtain the following inequality

$$\mathcal{L}_{\mathbf{u},k}(\mathbf{u}^k,.) - \mathcal{L}_{\mathbf{u},k}(\mathbf{u}^{k+1},.) \geq \frac{1+\rho_1}{2}\left\|\mathbf{u}^{k+1} - \mathbf{u}^k\right\|_2^2, \tag{3.70}$$

thanks to Proposition 19. It results in the Proposition (P-1).

3.7.1.2 Proof of Proposition (P-2)

The minimizer $\mathbf{s}^{k+1}$ defined in (3.18) satisfies

$$\mathcal{L}\left(\mathbf{s}^{k+1}, \mathbf{u}^{k+1}, \mathbf{r}^k, \mathbf{w}^k, \mathbf{e}^k\right) \leq \mathcal{L}\left(\mathbf{s}^k, \mathbf{u}^{k+1}, \mathbf{r}^k, \mathbf{w}^k, \mathbf{e}^k\right) - c_s\left\|\mathbf{s}^k - \mathbf{s}^{k+1}\right\|_2^2. \tag{P-2}$$

At the k-th iteration, the variable $\mathbf{s}$ is updated by minimizing the objective function $\mathcal{L}_{\mathbf{s},k}(\mathbf{s},.)$ in Eq. (3.15), as

$$\mathbf{s}^{k+1} = \underset{\mathbf{s}}{\operatorname{argmin}} \left[\mathcal{L}_{\mathbf{s},k}(\mathbf{s},.) = \rho\|\mathbf{s}\|_1 + \frac{\rho_1}{2}\left\|\mathbf{s} - \left(\mathbf{u}^{k+1} + \mathbf{r}^k\right)\right\|_2^2\right]. \tag{3.71}$$

We exploit that if given $\mathbf{u}^{k+1}$ and $\mathbf{r}^k$, then both functions of the ℓ_1-norm $\|\mathbf{s}\|_1$ and ℓ_2-norm $\left\|\mathbf{s} - (\mathbf{u}^{k+1} + \mathbf{r}^k)\right\|_2^2$ are convex, so the $\mathcal{L}_{\mathbf{s},k}(\mathbf{s},.)$ w.r.t. $\mathbf{s}$ is also convex. It is therefore that for any $\mathbf{s}^k, \mathbf{s}^{k+1} \in \mathcal{S}$, we always have

$$\mathcal{L}_{\mathbf{s},k}\left(\mathbf{s}^k,.\right) \geq \mathcal{L}_{\mathbf{s},k}\left(\mathbf{s}^{k+1},.\right) + \left\langle \mathbf{s}^k - \mathbf{s}^{k+1}, \nabla\mathcal{L}_{\mathbf{s},k}\left(\mathbf{s}^{k+1},.\right)\right\rangle + \frac{1}{2}\left\|\mathbf{s}^{k+1} - \mathbf{s}^k\right\|_2^2, \tag{3.72}$$

thanks to the Proposition 3.

Since $\mathbf{s}^{k+1} = \operatorname{argmin}_\mathbf{s} \mathcal{L}_{\mathbf{s},k}(\mathbf{s},.)$, the first derivative $\nabla\mathcal{L}_{\mathbf{s},k}(\mathbf{s}^{k+1},.) = \mathbf{0}$ and hence

$$\mathcal{L}_{\mathbf{s},k}\left(\mathbf{s}^k,.\right) \geq \mathcal{L}_{\mathbf{s},k}\left(\mathbf{s}^{k+1},.\right). \tag{3.73}$$

In other word, there always exists a nonnegative number $c_s \geq 0$ such that

$$\mathcal{L}_{\mathbf{s},k}\left(\mathbf{s}^k,.\right) \geq \mathcal{L}_{\mathbf{s},k}\left(\mathbf{s}^{k+1},.\right) + \frac{1}{2}\left\|\mathbf{s}^{k+1} - \mathbf{s}^k\right\|_2^2. \tag{3.74}$$

As a result, we have

$$\sum_{k=1}^K \frac{1}{2}\left\|\mathbf{s}^{k+1} - \mathbf{s}^k\right\|_2^2 \leq \sum_{i=1}^K \mathcal{L}_{\mathbf{s},k}\left(\mathbf{s}^k,.\right) - \mathcal{L}_{\mathbf{s},k}\left(\mathbf{s}^{k+1},.\right) = \mathcal{L}_{\mathbf{s},k}\left(\mathbf{s}^1,.\right) - \mathcal{L}_{\mathbf{s},k}\left(\mathbf{s}^{K+1},.\right). \tag{3.75}$$

Let $K \to \infty$, we then have $\sum_{k=1}^\infty \left\|\mathbf{s}^{k+1} - \mathbf{s}^k\right\|_2^2 < \infty$. It ends the proof of (P-2) and the second part of Lemma 1.

3.7.1.3   Proof of Proposition (P-3)

The minimizer $\mathbf{r}^{k+1}$ defined in (3.16) satisfies

$$\mathcal{L}\big(\mathbf{s}^{k+1}, \mathbf{u}^{k+1}, \mathbf{r}^{k+1}, \mathbf{w}^k, \mathbf{e}^k\big) \leq \mathcal{L}\big(\mathbf{s}^{k+1}, \mathbf{u}^{k+1}, \mathbf{r}^k, \mathbf{w}^k, \mathbf{e}^k\big) - c_r\big\|\mathbf{r}^k - \mathbf{r}^{k+1}\big\|_2^2. \quad \text{(P-3)}$$

Follow the $\mathbf{r}$-update in Eq. (3.16), it is easy to verify that

$$\begin{aligned}
\mathcal{L}\big(\mathbf{s}^{k+1}, \mathbf{u}^{k+1}, \mathbf{r}^{k+1}, \mathbf{w}^k, \mathbf{e}^k\big) &= \rho_1\big(\mathbf{r}^k + \mathbf{s}^{k+1} - \mathbf{u}^{k+1}\big)^\top\big(\mathbf{u}^{k+1} - \mathbf{s}^{k+1}\big) + A \\
&= \rho_1(\mathbf{r}^k)^\top\big(\mathbf{u}^{k+1} - \mathbf{s}^{k+1}\big) - \rho_1\big\|\mathbf{u}^{k+1} - \mathbf{s}^{k+1}\big\|_2^2 + A \\
&= \mathcal{L}\big(\mathbf{s}^{k+1}, \mathbf{u}^{k+1}, \mathbf{r}^k, \mathbf{w}^k, \mathbf{e}^k\big) - \rho_1\big\|\mathbf{r}^{k+1} - \mathbf{r}^k\big\|_2^2, \quad (3.76)
\end{aligned}$$

where $A = g(\mathbf{s}^{k+1}) + h(\mathbf{u}^{k+1}) + \frac{\rho_1}{2}\|\mathbf{u}^{k+1} - \mathbf{s}^{k+1}\|$. It implies the proposition (P-3).

3.7.1.4   Proof of Proposition (P-4)

The minimizer $\mathbf{w}^{k+1}$ defined in (3.27) satisfies

$$\mathcal{L}\big(\mathbf{s}^{k+1}, \mathbf{u}^{k+1}, \mathbf{r}^{k+1}, \mathbf{w}^{k+1}, \mathbf{e}^k\big) \leq \mathcal{L}\big(\mathbf{s}^{k+1}, \mathbf{u}^{k+1}, \mathbf{r}^{k+1}, \mathbf{w}^k, \mathbf{e}^k\big) - c_w\big\|\mathbf{w}^k - \mathbf{w}^{k+1}\big\|_2^2.$$
$$\text{(P-4)}$$

Denote $\mathbf{z} = \mathbf{P}_t(\mathbf{U}_t\mathbf{w} + \mathbf{s}^{k+1} - \mathbf{x}_t)$. In fact, the $\mathbf{w}$-update minimizes the smooth version of the objective function (3.23), as follows

$$\mathcal{L}_{\mathbf{z},k}(\mathbf{z}, .) = \sum_{i=1}^{n}\left[\big((\mathbf{z}(i)^2 + 1)^{1/2} - 1\big) + \frac{\rho_2}{2}\big((\mathbf{z}(i) - \mathbf{e}^k(i))^2 + 1)^{1/2} - 1\big)\right].$$
$$(3.77)$$

The first two derivatives of $\mathcal{L}_{\mathbf{z},k}(\mathbf{z}, .)$ are given by

$$\begin{aligned}
\nabla\mathcal{L}_{\mathbf{z},k}(\mathbf{z}, .) &= \Big[\mathbf{z}(1)(\mathbf{z}(1)^2 + 1)^{-1/2}, \ldots, \mathbf{z}(n)(\mathbf{z}(n)^2 + 1)^{-1/2}\Big]^\top \\
&\quad + \rho_2\Big[(\mathbf{z}(1) - \mathbf{e}^k(1))((\mathbf{z}(1) - \mathbf{e}^k(1))^2 + 1)^{-1/2}, \ldots, \\
&\qquad \ldots(\mathbf{z}(n) - \mathbf{e}^k(n))((\mathbf{z}(1) - \mathbf{e}^k(1))^2 + 1)^{-1/2}\Big]^\top, \quad (3.78)
\end{aligned}$$

and

$$\begin{aligned}
\nabla^2\mathcal{L}_{\mathbf{z},k}(\mathbf{z}, .) &= \mathtt{diag}\bigg(\Big[(\mathbf{z}(1)^2 + 1)^{-3/2}, \ldots, (\mathbf{z}(n)^2 + 1)^{-3/2}\Big]\bigg) \\
&\quad + \rho_2\mathtt{diag}\bigg(\Big[((\mathbf{z}(1) - \mathbf{e}^k(1))^2 + 1)^{-3/2}, \ldots, (\mathbf{z}(n) - \mathbf{e}^k(n))^2 + 1)^{-3/2}\Big]\bigg).
\end{aligned}$$
$$(3.79)$$

The Hessian matrix $\nabla^2 \mathcal{L}_{z,k}(\mathbf{z}, .)$ then satisfies $\rho_2 \mathbf{I} < \nabla^2 \mathcal{L}_{z,k}(\mathbf{z}, .) \leq (\rho_2 + 1)\mathbf{I}$. It is therefore that $\mathcal{L}_{z,k}(\mathbf{w}, .)$ is strongly convex and Lipschitz continuous. In other word, it implies that

$$\mathcal{L}(\mathbf{s}^{k+1}, \mathbf{u}^{k+1}, \mathbf{r}^{k+1}, \mathbf{w}^k, \mathbf{e}^k) - \mathcal{L}(\mathbf{s}^{k+1}, \mathbf{u}^{k+1}, \mathbf{r}^{k+1}, \mathbf{w}^{k+1}, \mathbf{e}^k) > \frac{\rho_2}{2}\|\mathbf{w}^k - \mathbf{w}^{k+1}\|_2^2. \tag{3.80}$$

which results in the Proposition (P-4), thanks to Proposition 19.

3.7.1.5   Proof of Proposition (P-5)

The minimizer $\mathbf{e}^{k+1}$ defined in (3.29) satisfies

$$\mathcal{L}(\mathbf{s}^{k+1}, \mathbf{u}^{k+1}, \mathbf{r}^{k+1}, \mathbf{w}^{k+1}, \mathbf{e}^{k+1}) \leq \mathcal{L}(\mathbf{s}^{k+1}, \mathbf{u}^{k+1}, \mathbf{r}^{k+1}, \mathbf{w}^{k+1}, \mathbf{e}^k) - c_e\|\mathbf{e}^k - \mathbf{e}^{k+1}\|_2^2. \tag{P-5}$$

Similarly, we also have $\mathcal{L}_{e,k}(\mathbf{e}, .)$ is strongly convex, i.e.,

$$\nabla^2 \mathcal{L}_{e,k}(\mathbf{e}, .) = \rho_2 \operatorname{diag}\Bigg(\Big[\big((\mathbf{z}^k(1) - \mathbf{e}(1))^2 + 1\big)^{-3/2}, \dots,$$
$$\big(\mathbf{z}^k(n) - \mathbf{e}(n))^2 + 1\big)^{-3/2}\Big]\Bigg). \tag{3.81}$$

Therefore we have

$$\mathcal{L}_{e,k}(\mathbf{e}^k, .) - \mathcal{L}_{e,k}(\mathbf{e}^{k+1}, .) \geq \frac{\rho_2}{2}\left\|\mathbf{e}^{k+1} - \mathbf{e}^k\right\|_2^2. \tag{3.82}$$

It ends the proof.

3.7.2   Proof of Proposition 2

To prove that $g_t(\mathbf{U})$ is strongly convex, we state the following facts: $g_t(\mathbf{U})$ is continuous and differentiable; its second derivative is a positive semi-definite matrix (i.e., $\nabla_{\mathbf{U}}^2 g_t(\mathbf{U}) \succeq m\mathbf{I}$); and the domain of $g_t(\mathbf{U})$ is convex. In order to satisfy the Lipschitz condition, we show that the first derivative of $g_t(\mathbf{U})$ is bounded.

Stage I: Prove that g_t is a strong convex function

We show that there exists a positive number m such that

$$\left|g_t(\mathbf{U}_{t+1}) - g_t(\mathbf{U}_t)\right| \geq m_1 \|\mathbf{U}_{t+1} - \mathbf{U}_t\|_F^2. \tag{3.83}$$

In particular, we state the two claims as follows:

(C-1): $g_t(\mathbf{U})$ is continuous and differentiable.

> *Proof.* Given two variables $\mathbf{A}, \mathbf{B} \in \mathcal{U}$ such that $\|\mathbf{A} - \mathbf{B}\|_F^2 < \gamma$ for some positive constant γ. It is easy to verify that there exists a positive number θ such that $|g_t(\mathbf{A}) - g_t(\mathbf{B})| < \theta$.

Thanks to the triangle inequality, we have the following inequality:

$$
\begin{aligned}
&\left| g_t(\mathbf{A}) - g_t(\mathbf{B}) \right| \\
&= \frac{1}{t} \left| \sum_{i=1}^{t} \beta^{t-i} \left\| \mathbf{P}_i(\mathbf{A}\mathbf{w}_i + \mathbf{s}_i - \mathbf{x}_i) \right\|_2^2 - \sum_{i=1}^{t} \beta^{t-i} \left\| \mathbf{P}_i(\mathbf{B}\mathbf{w}_i + \mathbf{s}_i - \mathbf{x}_i) \right\|_2^2 \right| \\
&\leq \frac{1}{t} \sum_{i=1}^{t} \beta^{t-i} \left\| \mathbf{P}_i(\mathbf{A} - \mathbf{B})\mathbf{w}_i \right\|_2^2 \leq \frac{1}{t} \sum_{i=1}^{t} \beta^{t-i} \left\| \mathbf{P}_i(\mathbf{A} - \mathbf{B}) \right\|_F^2 \left\| \mathbf{w}_i \right\|_2^2 \\
&\leq \frac{1}{t} \sum_{i=1}^{t} \beta^{t-i} \left\| \mathbf{A} - \mathbf{B} \right\|_F^2 \left\| \mathbf{w}_i \right\|_2^2 = \frac{\gamma}{t} \sum_{i=1}^{t} \beta^{t-i} \left\| \mathbf{w}_i \right\|_2^2 = \theta,
\end{aligned}
\tag{3.84}
$$

Therefore, the set of functions $\{g_t(\mathbf{U})\}_{t=1}^{\infty}$ is equicontinuous on $\mathcal{U}$.

Furthermore, for any $\mathbf{U}^*, \mathbf{H} \in \mathcal{U}$, we show that the following limit exists:

$$
\lim_{a \to 0} \frac{g_t(\mathbf{U}^* + a\mathbf{H}) - g_t(\mathbf{U}^*)}{a} = \lim_{a \to 0} \frac{1}{ta} \sum_{i=1}^{t} \beta^{t-i} \left(\left\| \mathbf{P}_i\big((\mathbf{U}^* + a\mathbf{H})\mathbf{w}_i + \mathbf{s}_i - \mathbf{x}_i\big) \right\|_2^2 \right.
$$
$$
\left. - \left\| \mathbf{P}_i(\mathbf{U}^*\mathbf{w}_i + \mathbf{s}_i - \mathbf{x}_i) \right\|_2^2 \right).
\tag{3.85}
$$

Specifically, let us denote $\mathbf{y}_i = \mathbf{P}_i(\mathbf{U}^*\mathbf{w}_i + \mathbf{s}_i - \mathbf{x}_i)$, the limit can be written as follows:

$$
\lim_{a \to 0} \frac{g_t(\mathbf{U}^* + a\mathbf{H}) - g_t(\mathbf{U}^*)}{a} = \lim_{a \to 0} \frac{1}{ta} \sum_{i=1}^{t} \beta^{t-i} \left(\left\| \mathbf{y}_i - a\mathbf{P}_i\mathbf{H}\mathbf{w}_i \right\|_2^2 - \left\| \mathbf{y}_i \right\|_2^2 \right)
$$
$$
= \lim_{a \to 0} \frac{1}{ta} \sum_{i=1}^{t} \beta^{t-i} \left(\left\| a\mathbf{P}_i\mathbf{H}\mathbf{w}_i \right\|_2^2 - 2a\langle \mathbf{u}_i, \mathbf{P}_i\mathbf{H}\mathbf{w}_i \rangle \right)
$$
$$
= \frac{-2}{t} \sum_{i=1}^{t} \beta^{t-i} \langle \mathbf{y}_i, \mathbf{P}_i\mathbf{H}\mathbf{w}_i \rangle < \infty.
\tag{3.86}
$$

As a result, the function $g_t(\mathbf{U})$ is differentiable and its first derivative $\nabla_{\mathbf{U}} g_t(\mathbf{U})$ can be given by

$$
\nabla_{\mathbf{U}} g_t(\mathbf{U}) = \frac{2}{t} \sum_{i=1}^{t} \beta^{t-i} \mathbf{P}_i(\mathbf{U}\mathbf{w}_i + \mathbf{s}_i - \mathbf{x}_i)\mathbf{w}_i^{\top}.
\tag{3.87}
$$

In the similar way, it is easy to verify that $\nabla_U g_t(U)$ is also continuous and the second derivative $\nabla_U^2 g_t(U)$ is given by

$$\nabla_U^2 g_t(U) = \frac{2}{t} \sum_{i=1}^{t} \beta^{t-i} P_i w_i w_i^\top.$$
(3.88)

(C-2): The second derivative $\nabla_U^2 g_t(U)$ is a positive-define matrix. For all $x \in \mathbb{R}^{p \times 1}$, we have

$$x^\top \nabla_U^2 g_t(U) x = \frac{2}{t} \sum_{i=1}^{t} \beta^{t-i} P_i (w_i^\top x)^\top (w_i^\top x)$$

$$= \frac{2}{t} \sum_{i=1}^{t} \beta^{t-i} P_i (w_i^\top x)^2 > 0, \quad \forall \beta, t > 0.$$
(3.89)

It implies that there always exist a positive constant m such that $\nabla_U^2 g_t(U) \geq mI$.

It follows to the claims (C-1), (C-2) and the assumptions showing that the domain of $g_t(U)$ is a convex set that $g_t(U_t)$ is strongly convex [132, Section 3.1.4].

Stage II: Prove that g_t is a Lipschitz function

$$\left| g_t(U_{t+1}) - g_t(U_t) \right| \leq m_2 \left\| U_{t+1} - U_t \right\|_F.$$
(3.90)

Let us denote $d_t(U) = g_t(U) - g_{t+1}(U)$. Since $U_t = \underset{U \in \mathcal{U}}{\arg\min}\, g_t(U)$, we exploit that $g_{t+1}(U_{t+1}) \leq g_{t+1}(U_t)$ and hence

$$g_t(U_{t+1}) - g_t(U_t) = g_t(U_{t+1}) - g_{t+1}(U_t) + g_{t+1}(U_t) - g_t(U_t)$$

$$\leq \Big(\underbrace{g_t(U_{t+1}) - g_{t+1}(U_{t+1})}_{d_t(U_{t+1})} \Big) - \Big(\underbrace{g_t(U_t) - g_{t+1}(U_t)}_{d_t(U_t)} \Big). \quad (3.91)$$

The first derivative of $d_t(U) = g_t(U) - g_{t+1}(U)$ is given by

$$\nabla_U d_t(U) = \nabla_U g_t(U) - \nabla_U g_{t+1}(U)$$

$$= \frac{1}{t} \sum_{i=1}^{t} \beta^{t-i} P_i (U w_i + s_i - x_i) w_i^\top$$

$$- \frac{1}{t+1} \sum_{i=1}^{t+1} \beta^{t+1-i} P_i (U w_i + s_i - x_i) w_i^\top.$$
(3.92)

Let $\mathbf{A}_t = \sum_{i=1}^{t} \beta^{t-i} \mathbf{P}_i \mathbf{U} \mathbf{w}_i \mathbf{w}_i^\top$ and $\mathbf{B}_t = \sum_{i=1}^{t} \beta^{t-i} \mathbf{P}_i (\mathbf{s}_i - \mathbf{x}_i)$, we can rewrite $\nabla_{\mathbf{U}} d_t(\mathbf{U})$ as

$$\nabla_{\mathbf{U}} d_t(\mathbf{U}) = \left(\frac{\mathbf{A}_t}{t} - \frac{\mathbf{A}_{t+1}}{t+1} \right) + \left(\frac{\mathbf{B}_t}{t} - \frac{\mathbf{B}_{t+1}}{t+1} \right). \tag{3.93}$$

Under the assumptions in Section 3.2.2, the subspace $\mathbf{U}$, outlier $\{\mathbf{s}_t\}$, signal $\{\mathbf{x}_t\}$ and coefficients $\{\mathbf{w}_t\}$ are bounded, then both $\mathbf{A}_t$ and $\mathbf{B}_t$ are bounded. It is therefore that

$$\|\nabla_{\mathbf{U}} d_t(\mathbf{U})\|_F \leq \left\| \frac{\mathbf{A}_t}{t} - \frac{\mathbf{A}_{t+1}}{t+1} \right\|_F + \left\| \frac{\mathbf{B}_t}{t} - \frac{\mathbf{B}_{t+1}}{t+1} \right\|_F \leq m_2 = \mathcal{O}(1/t). \tag{3.94}$$

Therefore $d_t(\mathbf{U})$ is Lipschiz with the constant m_2,

$$\frac{|d_t(\mathbf{U}_{t+1}) - d_t(\mathbf{U}_t)|}{\|\mathbf{U}_{t+1} - \mathbf{U}_t\|_F} \leq m_2, \text{ hence } \frac{|g_t(\mathbf{U}_{t+1}) - g_t(\mathbf{U}_t)|}{\|\mathbf{U}_{t+1} - \mathbf{U}_t\|_F} \leq m_2. \tag{3.95}$$

This ends the proof.

3.7.3   Proof of Lemma 2

We prove that our update rule is an approximate interpretation of Newton's method. Since the objective function g_t is strongly convex with respect to the variable $\mathbf{U}$, our algorithm can guarantee that the solution converges to the stationary point of the problem.

In order to estimate subspace, at each time instant t, we optimize the following minimization

$$\mathbf{u}^m = \underset{\mathbf{u}^m \in \mathbb{R}^{r \times 1}}{\operatorname{argmin}} \left[\tilde{f}_t(\mathbf{u}^m) = \sum_{i=1}^{t} \beta^{t-i} \mathbf{P}_i(m,m) \big(\mathbf{x}_i^{\mathrm{re}}(m) - \mathbf{w}_i^\top \mathbf{u}^m \big)^2 + \frac{\alpha}{2t} \|\mathbf{u}^m\|_2^2 \right]. \tag{3.96}$$

The first derivative of the objective function $\tilde{f}_t(\mathbf{u}^m)$ can be determined by

$$\nabla \tilde{f}_t(\mathbf{u}_{t-1}^m) = -2 \sum_{i=1}^{t} \beta^{t-i} \mathbf{P}_i(m,m) \big(\mathbf{x}_i^{\mathrm{re}}(m) - \mathbf{w}_i^\top \mathbf{u}_{t-1}^m \big) \mathbf{w}_i^\top + \frac{\alpha}{t} \mathbf{u}_{t-1}^m$$

$$= \nabla \tilde{f}_{t-1}(\mathbf{u}_{t-1}^m) - 2 \mathbf{P}_t(m,m) \big(\mathbf{x}_t^{\mathrm{re}}(m) - \mathbf{w}_t^\top \mathbf{u}_{t-1}^m \big) \mathbf{w}_t^\top + \frac{\alpha}{t} \big(\mathbf{u}_{t-1}^m - \mathbf{u}_{t-2}^m \big). \tag{3.97}$$

Since $\mathbf{u}_{t-1}^m = \operatorname{argmin}_{\mathbf{u}^m} \tilde{f}_{t-1}(\mathbf{u}^m)$, the derivative $\nabla \tilde{f}_{t-1}(\mathbf{u}_{t-1}^m) = \mathbf{0}$ and the Hessian at $\mathbf{u}_{t-1}^m$ is then given by

$$\mathcal{H}\tilde{f}_t(\mathbf{u}_{t-1}^m) = \nabla^2 \tilde{f}_t(\mathbf{u}_{t-1}^m) = 2 \sum_{i=1}^{t} \beta^{t-i} \mathbf{P}_i(m,m) \mathbf{w}_i \mathbf{w}_i^\top + \frac{\alpha}{t} \mathbf{I}. \tag{3.98}$$

Thanks to Newton's method [132], a rule for subspace update can be obtained as

$$\mathbf{u}_t^m = \mathbf{u}_{t-1}^m - \eta_t \big(\mathcal{H}\tilde{f}_t(\mathbf{u}_{t-1}^m)\big)^{-1}\nabla\tilde{f}_t(\mathbf{u}_{t-1}^m). \tag{3.99}$$

Let us denote $\mathbf{R}_t^m = \sum_{i=1}^{t}\beta^{t-i}\mathbf{P}_t(m,m)\mathbf{w}_i\mathbf{w}_i^\top + \alpha\big(\frac{1}{2t} - \frac{\beta_t}{2(t-1)}\big)\mathbf{I}$, we have

$$\mathcal{H}\tilde{f}_t(\mathbf{u}_{t-1}^m) = 2\mathbf{R}_t^m + \alpha\left(\frac{\beta_t}{2(t-1)} - \frac{1}{2t}\right)\mathbf{I}. \tag{3.100}$$

As a result, we can derive the inverse Hessian matrix easily as follows

$$\big(\mathcal{H}\tilde{f}_t(\mathbf{u}_{t-1}^m)\big)^{-1} = \frac{1}{2}\big(\mathbf{R}_t^m\big)^{-1}\left(\frac{O(1/t)}{2}\big(\mathbf{R}_t^m\big)^{-1} + \mathbf{I}\right)^{-1}. \tag{3.101}$$

When t is large enough, the term $\big(\frac{O(1/t)}{2}(\mathbf{R}_t^m)^{-1} + \mathbf{I}\big)^{-1} \approx \mathbf{I} + O\big(\frac{1}{t}\big)$. It is therefore that the step size can be approximated by

$$\big[\mathcal{H}\tilde{f}_t(\mathbf{u}_{t-1}^m)\big]^{-1}\nabla\tilde{f}_t(\mathbf{u}_{t-1}^m) = -\mathbf{P}_t(m,m)\big(\mathbf{x}_t^{\text{re}}(m) - \mathbf{w}_t^\top\mathbf{u}_{t-1}^m\big)\big(\mathbf{R}_t^m\big)^{-1}\mathbf{w}_t + O\big(1/t\big). \tag{3.102}$$

It implies that $\mathbf{u}_t^m$ can be updated by the following recursive update rule

$$\mathbf{u}_t^m = \mathbf{u}_{t-1}^m + \eta_t\mathbf{P}_t(m,m)\big(\mathbf{x}_t^{\text{re}}(m) - \mathbf{w}_t^\top\mathbf{u}_{t-1}^m\big)\big(\mathbf{R}_t^m\big)^{-1}\mathbf{w}_t, \tag{3.103}$$

which is already defined in Eq. (3.14). In other word, the $\mathbf{u}_t^m$ generated by our algorithm can converge to the stationary point of $\tilde{f}_t(\mathbf{u}^m)$.

Note that, the properties of the objective functions and assumptions we made in Section 3.2.2 can guarantee the method will converge in practice. In particular, the objective functions $\tilde{g}_t(\mathbf{U})$ as well as $\tilde{f}_t(\mathbf{u})$ and their first derivatives are continuously differentiable which can avoid derivative issues in Newton's method. In addition, the starting points in our algorithm are always chosen at random. Further, since the objective functions $\{\tilde{g}_t(\mathbf{U})\}_{t=1}^{\infty}$ are always positive, PETRELS-ADMM can ignore the cases when their roots approach to zero asymptotically. To sum up, the solution $\mathbf{U}_t$ generated by PETRELS-ADMM will converge to the stationary point of the function $\tilde{g}_t(\mathbf{U})$.

The second part of the Lemma 3.7.3 can be easy to verify. Since $g_t(\mathbf{U}_t)$ is strongly convex and Lipschitz function as proved in Proposition 2, we have the following inequality

$$m_1\|\mathbf{U}_{t+1} - \mathbf{U}_t\|_F^2 \leq \big|g_t(\mathbf{U}_{t+1}) - g_t(\mathbf{U}_t)\big| \leq m_2\|\mathbf{U}_{t+1} - \mathbf{U}_t\|_F$$
$$\Leftrightarrow \|\mathbf{U}_{t+1} - \mathbf{U}_t\|_F\left(\|\mathbf{U}_{t+1} - \mathbf{U}_t\|_F - \frac{m_2}{m_1}\right) \leq 0 \Leftrightarrow \|\mathbf{U}_t - \mathbf{U}_{t+1}\|_F \leq \frac{m_2}{m_1}. \tag{3.104}$$

Note that the positive number $m_2 = O(1/t)$ is already given in the Appendix 3.7.2, so it ends the proof.

3.7.4 Proof of Lemma 3

Inspired of the result of convergence analysis for online sparse coding framework in [120, Proposition 2], we derive the convergence of $g_t(\mathbf{U}_t)$ in the similar way. In particular, we first denote the nonnegative stochastic process $\{u_t\}$ as follows

$$u_t \overset{\Delta}{=} g_t(\mathbf{U}_t) \geq 0, \tag{3.105}$$

and then prove that it is a quasi-martingale, i.e., we have to prove the sum of the positive difference of $\{u_t\}_{t=1}^{\infty}$ is bounded,

$$\sum_{t=1}^{\infty} \left| \mathbb{E}[u_{t+1} - u_t] \right| < +\infty \quad a.s. \tag{3.106}$$

We can express $g_{t+1}(\mathbf{U}_t)$ with respect to $g_t(\mathbf{U}_t)$ as follows

$$g_{t+1}(\mathbf{U}_t) = \frac{1}{t+1} \sum_{i=1}^{t+1} \beta^{t+1-i} \left\| \mathbf{P}_i(\mathbf{U}_t \mathbf{w}_i + \mathbf{s}_i - \mathbf{x}_i) \right\|_2^2 + \rho \left\| \mathbf{s}_i \right\|_1$$

$$= \left(\frac{\beta}{t+1} \sum_{i=1}^{t} \beta^{t-i} \left\| \mathbf{P}_i(\mathbf{U}_t \mathbf{w}_i + \mathbf{s}_i - \mathbf{x}_i) \right\|_2^2 + \rho \left\| \mathbf{s}_i \right\|_1 \right)$$

$$+ \left(\frac{1}{t+1} \left(\left\| \mathbf{P}_{t+1} \mathbf{U}_t + \mathbf{s}_{t+1} - \mathbf{x}_{t+1} \right\|_2^2 + \rho \left\| \mathbf{s}_{t+1} \right\|_1 \right) \right)$$

$$= \frac{\beta t}{t+1} g_t(\mathbf{U}_t) + \frac{1}{t+1} \ell(\mathbf{U}_t, \mathbf{P}_{t+1}, \mathbf{x}_{t+1}). \tag{3.107}$$

Since $\mathbf{U}_{t+1} = \operatorname{argmin}_{\mathbf{U}} g_{t+1}(\mathbf{U})$, we have the fact $g_{t+1}(\mathbf{U}_{t+1}) - g_{t+1}(\mathbf{U}_t) \leq 0$, $f_t(\mathbf{U}_t) \leq g_t(\mathbf{U}_t)$, and hence

$$u_{t+1} - u_t = g_{t+1}(\mathbf{U}_{t+1}) - g_t(\mathbf{U}_t) = \underbrace{g_{t+1}(\mathbf{U}_{t+1}) - g_{t+1}(\mathbf{U}_t)}_{\leq 0} + g_{t+1}(\mathbf{U}_t) - g_t(\mathbf{U}_t)$$

$$\leq g_{t+1}(\mathbf{U}_t) - g_t(\mathbf{U}_t) = \frac{1}{t+1} \ell(\mathbf{U}_t, \mathbf{P}_{t+1}, \mathbf{x}_{t+1}) - \frac{t(1-\beta)+1}{t+1} g_t(\mathbf{U}_t). \tag{3.108}$$

It is therefore that

$$\mathbb{E}[u_{t+1} - u_t] \leq \frac{\mathbb{E}\left[\ell(\mathbf{U}_t, \mathbf{P}_{t+1}, \mathbf{x}_{t+1}) - (t(1-\beta)+1) g_t(\mathbf{U}_t) \right]}{t+1}$$

$$\leq \frac{\mathbb{E}\left[\ell(\mathbf{U}_t, \mathbf{P}_{t+1}, \mathbf{x}_{t+1}) - g_t(\mathbf{U}_t) \right]}{t+1} \leq \frac{\mathbb{E}\left[\ell(\mathbf{U}_t, \mathbf{P}_{t+1}, \mathbf{x}_{t+1}) \right] - f_t(\mathbf{U}_t)}{t+1}$$

$$= \frac{\mathbb{E}\left[f(\mathbf{U}_t) - f_t(\mathbf{U}_t) \right]}{t+1} = \underbrace{\left(\mathbb{E}\left[\sqrt{t}\big(f(\mathbf{U}_t) - f_t(\mathbf{U}_t)\big) \right] \right)}_{\mathbb{E}[G_t(\mathbf{U}_t)]} \underbrace{\left(\frac{1}{\sqrt{t}(t+1)} \right)}_{a_t}, \tag{3.109}$$

because of $f_t(\mathbf{U}_t) \leq g_t(\mathbf{U}_t)$ and $\mathbb{E}[\ell(\mathbf{U}_t, \mathbf{P}_{t+1}, \mathbf{x}_t)] = f(\mathbf{U}_t)$. In parallel, we exploit that $G_t(\mathbf{U}_t) = \sqrt{t}(f(\mathbf{U}_t) - f_t(\mathbf{U}_t))$ is the scaled and centered version of the empirical measure, which converges in distribution to a normal random variable, thanks to the center limit theorem. Hence $\mathbb{E}[\sqrt{t}(f(\mathbf{U}_t) - f_t(\mathbf{U}_t))]$ is bounded with a constant α. Then, the sum of the positive difference of $\mathbf{u}_t$ becomes

$$\sum_{t=1}^{\infty} |\mathbb{E}[u_{t+1} - u_t]| < \sum_{t=1}^{\infty} \frac{\alpha}{\sqrt{t}(t+1)}. \tag{3.110}$$

Furthermore, let us consider the convergence of the sum $\displaystyle\sum_{t=1}^{+\infty} \frac{\alpha}{\sqrt{t}(t+1)}$. We use the Cauchy-MacLaurin integral test [133] for convergence, as

$$\int_{t=1}^{+\infty} \frac{\alpha}{\sqrt{t}(t+1)} dt = \int_{x=1}^{+\infty} \frac{\alpha}{(x^2+1)} dx$$
$$= \alpha \arctan(x)|_1^{+\infty} = \alpha\big(\arctan(\infty) - \arctan(1)\big) < \infty. \tag{3.111}$$

In other words, since the sum of $\mathbf{a}_t$ convergences, hence $\sum_{t=1}^{\infty} \mathbb{E}[u_{t+1} - u_t] < \infty$. We complete the proof.

3.7.5   Proof of Lemma 4

We investigate the convergence of a surrogate sequence $\left\{(g_t(\mathbf{U}_t) - f_t(\mathbf{U}_t))\frac{1}{t+1}\right\}$ as follows

$$\frac{g_t(\mathbf{U}_t) - f_t(\mathbf{U}_t)}{t+1} = u_t - u_{t+1} + \underbrace{g_{t+1}(\mathbf{U}_{t+1}) - g_{t+1}(\mathbf{U}_t)}_{\leq 0} + \underbrace{\frac{t(\beta-1)}{t+1} g_t(\mathbf{U}_t)}_{\leq 0}$$

$$+ \frac{\ell(\mathbf{U}_t, \mathbf{P}_{t+1}, \mathbf{x}_{t+1}) - f_t(\mathbf{U}_t)}{t+1}$$

$$\leq \underbrace{u_t - u_{t+1}}_{(S-1)} + \underbrace{\frac{\ell(\mathbf{U}_t, \mathbf{P}_{t+1}, \mathbf{x}_{t+1}) - f_t(\mathbf{U}_t)}{t+1}}_{(S-2)}, \tag{3.112}$$

because of $u_t = g_t(\mathbf{U}_t)$ and $\lambda \leq 1$. Note that, (S-1) − (S-2) converge almost surely:

- The sequence $\mathbb{E}[u_t - u_{t+1}]$ converges almost surely as proved in Lemma 3.

- The sequence (S-2) also converges, thanks to the fact $\mathbb{E}\big[\ell(\mathbf{U}_t, \mathbf{P}_{t+1}, \mathbf{x}_{t+1})\big] = f(\mathbf{U}_t)$ and the convergence of $\frac{\mathbb{E}\big[f(\mathbf{U}_t) - f_t(\mathbf{U}_t)\big]}{t+1}$ as mentioned in the appendix 3.7.4.

It is therefore that the sequence $\left\{(g_t(\mathbf{U}_t) - f_t(\mathbf{U}_t))\frac{1}{t+1}\right\}$ converges almost surely, i.e.,

$$\sum_{t=0}^{+\infty} \left(g_t(\mathbf{U}_t) - f_t(\mathbf{U}_t)\right)\frac{1}{t+1} < +\infty. \tag{3.113}$$

On the other hand, the real sequence $\{\frac{1}{t+1}\}$ diverges, $\sum_{t=0}^{+\infty} \frac{1}{t+1} = \infty$. It implies that $g_t(\mathbf{U}_t) - f_t(\mathbf{U}_t)$ convergences, thanks to the Proposition 7.

Technical Propositions

Here, we provide the following propositions which help us to derive several important results in our proofs.

Proposition 3 ([134]) *The function f is strongly convex if and only if for all $\mathbf{u}, \mathbf{v} \in \boldsymbol{dom}(f)$ we always have*

$$f(\mathbf{v}) - f(\mathbf{u}) - \frac{1}{2}\|\mathbf{v} - \mathbf{u}\|_2^2 \geq \langle \mathbf{v} - \mathbf{u}, \boldsymbol{\theta} \rangle, \quad \forall \boldsymbol{\theta} \in \partial f(\mathbf{u}).$$

Proposition 4 ([132]) *The function f is m-strongly convex, with a constant m if and only if for all $\mathbf{u}, \mathbf{v} \in \boldsymbol{dom}(f)$ we always have*

$$\left|f(\mathbf{v}) - f(\mathbf{u})\right| \geq \frac{m}{2}\|\mathbf{v} - \mathbf{u}\|_2^2.$$

Proposition 5 ([132]) *Every norm on $\mathbb{R}^n$ is convex and the sum of convex functions is convex.*

Proposition 6 ([135]) *The Huber penalty function replaces the ℓ_1-norm $\|\mathbf{x}\|_1$, $\mathbf{x} \in \mathbb{R}^n$ is given by the sum $\sum_{i=1}^n f_\mu^{Hub}(x(i))$, where*

$$f_\mu^{Hub}(x(i)) = \begin{cases} \frac{x(i)^2}{2\mu}, & |x(i)| \leq \mu, \\ |x(i)| - \mu/2, & |x| > \mu. \end{cases}$$

There exists a smooth version of the Huber function f_μ^{Hub}, which has derivatives of all degrees,

$$\psi_\mu(\mathbf{x}) = \sum_{i=1}^n \left((x(i)^2 + \mu^2)^{1/2} - \mu\right).$$

and the first derivative of the pseudo-Huber function ψ_μ is defined by

$$\nabla\psi_\mu(\mathbf{x}) = \left[x(1)(x(1)^2 + \mu^2)^{-1/2}, \ldots, x(n)(x(n)^2 + \mu^2)^{-1/2}\right]^\top.$$

Proposition 7 ([136, Proposition 1.2.4]) *Let $\{a_t\}_{t=1}^{\infty}$ and $\{b_t\}_{t=1}^{\infty}$ be two nonnegative sequences such that $\sum_{i=1}^{\infty} a_i = \infty$ and $\sum_{i=1}^{\infty} a_i b_i < \infty$, $|b_{t+1} - b_t| < K a_t$ with some constant K, then $\lim\limits_{t \to \infty} b_t = 0$ or $\sum_{i=1}^{\infty} b_i < \infty$.*

Proposition 8 *If $\{f_t\}_{t \geq 1}$ and $\{g_t\}_{t \geq 1}$ are sequences of bounded functions which converge uniformly on a set $\mathcal{E}$, then $\{f_t + g_t\}_{t \geq 1}$ and $\{f_t g_t\}_{t \geq 1}$ converge uniformly on $\mathcal{E}$.*

Sparse Subspace Tracking in High Dimensions

4

In recent years, sparse subspace tracking has attracted increasing attention in the signal processing community. In this chapter, we propose a new provable effective method called OPIT for tracking the sparse principal subspace of data streams over time. Particularly, OPIT introduces a new adaptive variant of power iteration with space and computational complexity linear to the data dimension. In addition, a new column-based thresholding operator is developed

to regularize the subspace sparsity. Utilizing both advantages of power iteration and thresholding operation, OPIT is capable of tracking the underlying subspace in both classical regime and high dimensional regime. We also present a theoretical result on its convergence to verify its consistency in high dimensions. Several experiments are carried out on both synthetic and real data to demonstrate the tracking ability of OPIT.

4.1 Introduction

Subspace tracking (ST) is an essential and fundamental problem in signal processing with various applications to sensor array processing, wireless communication, and image/video processing, to name a few [20]. It corresponds to the problem of tracking a low-rank subspace that can represent data streams. Most of subspace tracking methods are designed to estimate the underlying subspace from the sample covariance matrix (SCM). We refer the reader to [20, 21, 26] for good surveys on standard and robust ST algorithms.

Recently, many rigorous evidences and theoretical results in random matrix theory (e.g. [22–24]) indicated that the SCM is not a good estimator of the actual covariance matrix in high-dimension, low-sample-size (HDLSS) contexts where datasets are massive in both dimension n and sample size T, and typically $n/T \rightarrow c \in (0, \infty]$. In most online applications, this regime is indeed more realistic and relevant than the classical one where n is fixed and $T \rightarrow \infty$. It is mainly due to the time variation of (big) data streams in nonstationary environments where the underlying data distribution changes with time.[1] Accordingly, the data covariance matrix and the principal subspace are time varying too, and thus, the "effective" window length which defines actual data samples under processing is limited. Meanwhile, modern data streams are originally associated with high dimensionality [2]. This leads to the case in which the data dimension n is comparable or even larger than the actual number of snapshots under consideration T.

Without further structural knowledge about the data, subspace tracking algorithms turn out to be inconsistent in such a regime. Interestingly, the consistency of covariance estimation can be guaranteed under suitably structured sparsity regularizations [138–142]. Therefore, sparse subspace estimation and tracking have recently gained much attention in the signal processing community. In the literature, several good methods have been proposed for sparse subspace estimation, see [101, 143–145] for examples and [49, 146] for comprehensive surveys. However, in an adaptive (online) setting, there have been only few studies on sparse subspace tracking (SST) so far.

---

[1]This phenomenon is often referred to as concept drift or dataset shift in data mining and machine learning [137].

4.1.1 Related Works

As mentioned before, some online algorithms have been introduced for sparse subspace tracking [26]. A few of them are based on a two-stage approach in which one first utilizes a standard ST algorithm to estimate the underlying subspace and then seek a sparse basis of the estimation under some sparsity criteria. Particularly in [97, 98, 104], several variants of OPAST and FAPI were proposed to track the sparse principal subspace. Another good approach is to regularize the objective function that aims at accounting for the sparse basis. In [95], the authors modified the objective function of PAST by adding a ℓ_1-norm regularization term on the subspace matrix and then proposed a new robust variant of PAST called ℓ_1-PAST to optimize it. Similar to ℓ_1-PAST, the authors in [147] also introduced another adaptive algorithm using ℓ_1-norm minimization called SPCAur for sparse subspace tracking. SPCAur adopts the stochastic gradient descent on Grassmann manifolds and it is capable of tracking the underlying sparse subspace from incomplete observations. In [96], a Bayesian-based algorithm called OVBSL was proposed to deal with the sparsity constraint on the subspace matrix. An advantage of OVBSL is that it is fully automated, i.e., no finetuning parameter is required. However, these algorithms are only effective in the classical regime where the sample size is much larger than the dimension, i.e., $n/T \rightarrow 0$ asymptotically.

Through the lens of machine learning and statistics, SST is generally referred to as the problem of online sparse PCA which often emphasizes the leading eigenvectors. In [93], the authors proposed an extended version of the Oja algorithm for online sparse PCA, namely OIST. Its convergence, steady-state, and phase transition were also derived to investigate the use of OIST in high dimensions. OIST is, however, designed only for rank-1 sparse subspaces. In [94], another online sparse PCA algorithm (SSPCA) was proposed and could deal with rank-r subspaces. Specifically, this algorithm uses a simple row truncation operator, which sets rows whose scores are smaller than a threshold to zero, for tracking the sparse principal subspace over time. However, this truncation operator is only designed for subspaces with a row-sparse support (i.e. all eigenvectors must share the same sparsity patterns) which may not always meet in practice. Indeed, it turns out to be ineffective for a sparse subspace with another support (e.g. elementwise sparsity). Its performance in terms of estimation accuracy is typically lower than other SST algorithms, see Fig. 4.4 and Fig. 4.5 for illustration.

It is worth noting that algorithms in [98, 104], OIST [93], and SSPCA [94] can be viewed as online variants of a classical method for principal subspace estimation, namely power iteration (PI). In the literature, there exist other power-based subspace trackers and they can be broadly categorized into the following classes: Oja-types [148, 149], Natural Power (NP)-types [150, 151],

Data Projection Method (DPM)-types [152, 153], and Approximated PI (API)-types [154, 155]. Specifically, all of them are designed for tracking the principal subspace of the SCM which is, however, not a good estimator of the true data covariance matrix in high dimensions. Accordingly, they turn out to be inconsistent estimators in the HDLSS regime.

In parallel, recent years have also witnessed considerable research advances on robust ST (RST) which aims to track the underlying subspace in the presence of data corruption [21, 26, 156]. For example, several RST algorithms were developed to handle sparse outliers, such as Grassmannian Robust Adaptive Subspace Tracking Algorithm (GRASTA) [157], Parallel Subspace Estimation and Tracking by Recursive Least Squares (PETRELS)-types [25, 62], and Recursive Projected Compressive Sensing (ReProCS)-types [63, 64]. To deal with impulsive noises, three potential approaches are robust statistics [82, 158], adaptive Kalman filtering [84, 87], and weighted RLS [62, 159]. Very recently, α-divergence was specifically exploited to bolster the tracking ability of the well-known PAST and FAPI trackers in noisy and contaminated environments [160, 161]. However, none of them is designed for subspace tracking in the HDLSS context.

4.1.2 Contribution and Significance

In this chapter, we introduce a new provable adaptive algorithm called OPIT (OPIT stands for Online Power Iteration via Thresholding) for sparse subspace tracking. OPIT takes both advantages of power iteration and thresholding methods, and hence offers several appealing features over the state-of-the-art SST/online sparse PCA algorithms.

First, OPIT belongs to the class of power methods, and thus its convergence rate is highly competitive compared to other SST algorithms, especially in the high SNR regime. Unlike the two SST algorithms based on power methods (i.e. OIST and SSPCA), OPIT utilizes old observations efficiently in a recursive way and still operates with linear space complexity. Accordingly, OPIT could obtain not only a faster convergence rate but also a better subspace estimation accuracy than OIST and SSPCA. Compared to OIST which is limited to tracking rank-1 sparse subspaces, OPIT has the capability of tracking rank-r subspaces over time. Compared to SSPCA which is useful for only subspaces with row-sparse supports, OPIT offers an effective subspace tracker which can deal with more generalized sparsity supports than SSPCA, thanks to a new thresholding operator to deal with subspace sparsity. In particular, we propose to apply column-based thresholding instead of row-based thresholding as in SSPCA. With this operator, OPIT has a great potential for handling several sparsity supports such as row-sparse, elementwise-sparse, and local region-sparse.

Different from the existing two-stage SST algorithms, OPIT has ability to track the sparse principal subspace with high accuracy in both the classical regime and the HDLSS regime. Theoretically, the subspaces derived from the two-stage algorithms are identical to those obtained by the corresponding standard ST algorithms (e.g. OPAST and FAPI) used in their first stage. It is due to the fact that the subspace spanned by a full rank matrix remains unchanged after any rotation. Accordingly, they still suffer the limitation of the SCM in the HDLSS regime. By contrast, our OPIT algorithm aims to track the underlying sparse subspace from a thresholded SCM. Simulation results indicate that OPIT provides a much better subspace estimation accuracy than the two-stage SST algorithms in high dimensions. More importantly, as indicated later in our theoretical analysis, the convergence of OPIT with the thresholding operation can be guaranteed under certain conditions.

In addition, OPIT is flexible and very adaptable for different scenarios. In particular, we can adjust its procedure for dealing with multiple incoming data streams. This feature is useful for application areas wherein block processing is required, i.e., a block of data samples is processed and analysed at one time. Next, it is easy to introduce regularization parameters into OPIT in order to regularize its performance in non-standard environments. Specifically, we can use a forgetting factor to discount the impact of distant observations as well as facilitate the tracking ability of OPIT in dynamic environments. Moreover, we can recast its update rule into a column-wise update. Thanks to the deflation transformation, we particularly derive a fast variant of OPIT called OPITd with lower complexity of both computation and memory storage. This variant is fast and useful for tracking high-dimension and large-scale data streams residing in a low-dimensional space. Last but not least, OPIT belongs to the class of provable subspace tracking algorithms in which its convergence is guaranteed. Under certain conditions, OPIT can achieve an ϵ-relative-error approximation with high probability when the number of observations is large enough.

4.1.3   Organization and Notations

The rest of the chapter is organized as follows. Section 4.2 formulates the SST problem. Section 7.3.2 presents the proposed OPIT algorithm and its variant OPITd while Section 4.4 establishes its convergence analysis. Section 4.5 provides several experiments to demonstrate performance of the proposed algorithms in comparison with the state-of-the-art algorithms. Section 4.6 concludes the chapter.

4.2 Problem Formulation

Assume that at time t, we collect a data sample $\mathbf{x}_t \in \mathbb{R}^{n \times 1}$ satisfying the signal model

$$\mathbf{x}_t = \boldsymbol{\ell}_t + \mathbf{n}_t. \tag{4.1}$$

Here, $\boldsymbol{\ell}_t \in \mathbb{R}^{n \times 1}$ is a low-rank signal living in a subspace[2] spanned by a sparse matrix $\mathbf{A}^{n \times r}$ with $r < n$ (i.e. $\boldsymbol{\ell}_t = \mathbf{A}\mathbf{w}_t$, where $\mathbf{w}_t \in \mathbb{R}^{r \times 1}$ is a weight vector) and $\mathbf{n}_t \in \mathbb{R}^{n \times 1}$ is an additive spatially white noise vector independent of $\boldsymbol{\ell}_t$. Sparse subspace tracking problem can be stated as follows:

> **Sparse Subspace Tracking**: *Given a set of data streams* $\{\mathbf{x}_t\}_{t=1}^T$, *we aim to estimate a sparse principle subspace* $\mathbf{A}_t$ *that compactly represents the span of signals* $\{\boldsymbol{\ell}_t\}_{t=1}^T$.

Generally, the underlying subspace can be estimated from the spectral analysis of the actual covariance matrix

$$\mathbf{C} = \mathbb{E}\{\mathbf{x}_t\mathbf{x}_t^\top\} = \mathbf{A}\mathbb{E}\{\mathbf{w}_t\mathbf{w}_t^\top\}\mathbf{A}^\top + \mathbb{E}\{\mathbf{n}_t\mathbf{n}_t^\top\}. \tag{4.2}$$

Without loss of generality, we suppose that $\mathbf{C}$ has the form $\mathbf{C} = \sigma_x^2\mathbf{A}\mathbf{A}^\top + \sigma_n^2\mathbf{I}_n$ where $\mathbb{E}\{\mathbf{w}_t\mathbf{w}_t^\top\} = \sigma_x^2\mathbf{I}_r$ and $\mathbb{E}\{\mathbf{n}_t\mathbf{n}_t^\top\} = \sigma_n^2\mathbf{I}_n$. Applying eigenvalue decomposition (EVD) on $\mathbf{C}$ yields

$$\mathbf{C} \overset{\text{EVD}}{=} \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top = \begin{bmatrix} \mathbf{U}_s & \mathbf{U}_n \end{bmatrix} \begin{bmatrix} \boldsymbol{\Lambda}_s & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda}_n \end{bmatrix} \begin{bmatrix} \mathbf{U}_s^\top \\ \mathbf{U}_n^\top \end{bmatrix}. \tag{4.3}$$

Here, $\boldsymbol{\Lambda} \in \mathbb{R}^{n \times n}$ is a diagonal matrix whose diagonal elements are eigenvalues of $\mathbf{C}$ sorted in decreasing order and $\mathbf{U} \in \mathbb{R}^{n \times n}$ contains the corresponding eigenvectors. Accordingly, $\mathbf{U}_s \in \mathbb{R}^{n \times r}$ and $\mathbf{U}_n \in \mathbb{R}^{n \times (n-r)}$ represent the principal subspace and the minor subspace of $\mathbf{C}$, respectively. The orthogonal projection matrix of the sparse principal subspace is unique (i.e., $\mathbf{U}_s\mathbf{U}_s^\top = \mathbf{A}\mathbf{A}^\#$), so $\mathbf{A}$ can be obtained as $\mathbf{A} = \mathbf{U}_s\mathbf{Q}^*$ with

$$\mathbf{Q}^* = \underset{\mathbf{Q} \in \mathbb{R}^{r \times r}}{\arg\min} \|\mathbf{U}_s\mathbf{Q}\|_0 \text{ s.t. } \mathbf{Q} \text{ is full-rank}, \tag{4.4}$$

where $\|.\|_0$ promotes the sparsity on $\mathbf{A}$. In several applications, we often emphasize the principal subspace rather than its specific basis, such as dimensionality reduction [162] and array processing [107]. In this work, our main

---

[2]In an adaptive scheme, the matrix $\mathbf{A}$ may be slowly varying with time, i.e., $\mathbf{A} = \mathbf{A}_t$. Our algorithm is capable of successfully estimating the subspace as well as tracking its variation along the time.

objective is to track the principal (signal) subspace of $\mathbf{A}$ while the sparsifying step (4.4) is optional.

Most state-of-the-art SST algorithms estimate the principal subspace of the sample covariance matrix $\mathbf{C}_T = 1/T \sum_{t=1}^{T} \mathbf{x}_t \mathbf{x}_t^\top$ [26]. However, in a high-dimensional regime where $n/T \nrightarrow 0$ a.s., $\mathbf{C}_T$ is not a good estimator of $\mathbf{C}$. This limitation in an adaptive scheme is not necessarily due to a data shortage but to the time variation which forces us to use a limited window of time instead of all the data. Particularly, it has been shown that $\mathbf{C}_T$ is not a consistent estimate of $\mathbf{C}$ in the HDLSS regime, e.g. [163–165]. As a result, most of SST algorithms are not good in high dimensions, as illustrated in Fig. 4.5.

On the other hand, under certain conditions, it is proved in [138, 166] that

$$\left\| \mathbf{C} - \tau(\mathbf{C}_T) \right\|_2 \to 0 \quad \text{a. s. as } T \to \infty, \tag{4.5}$$

where $\tau(.)$ is an appropriate thresholding operator. Thanks to (4.5), in the next section, we derive a novel adaptive (online) algorithm based on power iteration and thresholding technique that is capable of tracking the sparse principal subspace in both the classical regime and the HDLSS regime.

4.3   Proposed Methods

In this section, a novel effective algorithm using thresholding is developed for sparse subspace tracking. This algorithm is dubbed as OPIT which stands for Online Power Iteration via Thresholding. We next derive a fast variant of OPIT called OPITd with lower complexity, thanks to the deflation transformation. Some remarks on OPIT and OPITd are discussed in the following subsection.

4.3.1   OPIT Algorithm

We first recall the main steps of the standard power iteration (PI) method on which we primarily leverage in order to develop our OPIT algorithm, for computing the dominant eigenvectors of $\mathbf{C}_t$. At the ℓ-th iteration, PI particularly updates (i) $\mathbf{S}_\ell \leftarrow \mathbf{C}_t \mathbf{U}_{\ell-1}$ and (ii) $\mathbf{U}_\ell \leftarrow \mathrm{QR}(\mathbf{S}_\ell)$ be the Q-factor of QR factorization of $\mathbf{S}_\ell$. PI starts from an initial matrix $\mathbf{U}_0 \in \mathbb{R}^{n \times r}$ and returns an orthonormal matrix $\mathbf{U}_L$ where L is the number of iterations [20].

In an adaptive scheme, the iteration step of PI can coincide with the data collection in time. At time t, the sample covariance matrix $\mathbf{C}_t$ can be recursively updated by: $\mathbf{R}_t = \mathbf{R}_{t-1} + \mathbf{x}_t \mathbf{x}_t^\top$ and $\mathbf{C}_t = t^{-1} \mathbf{R}_t$. As streaming data can vary with time, we propose to use a forgetting factor β ($0 < \beta \leq 1$) to discount the impact of old observations exponentially. The underlying subspace

INPUT: $\{\mathbf{x}_i\}_{i=1}^T, \mathbf{x}_i \in \mathbb{R}^{n \times 1}$, target rank r, a forgetting factor $0 < \beta \leq 1$, window of length $W \geq 1$, and a thresholding factor k

$$k = \begin{cases} \lfloor (1 - \omega_{sparse})n \rfloor & \text{if } \omega_{sparse} \text{ is given,} \\ \lfloor 10r \log n \rceil & \text{if } \omega_{sparse} \text{ is unknown,} \end{cases}$$

where ω_{sparse} is the sparsity level of the sparse basis.

INITIALIZATION: $\mathbf{U}_0 = \text{randn}(n, r), \mathbf{S}_{0,\mathcal{F}} = \mathbf{0}_{n \times r}, \mathbf{E}_0 = \mathbf{0}_{r \times r}$

MAIN PROGRAM:

PROCEDURE
for $t = 1, 2, \ldots, T/W$ do
    $\mathbf{X}_t = [\mathbf{x}_{(t-1)W+1}, \ldots, \mathbf{x}_{tW}]$ // Data collection
    $\mathbf{Z}_t = \mathbf{U}_{t-1}^\top \mathbf{X}_t$
    $\mathbf{S}_t = \beta \frac{(t-1)W}{tW} \mathbf{S}_{t-1} \mathbf{E}_{t-1} + \frac{1}{tW} \mathbf{X}_t \mathbf{Z}_t^\top$
    $\hat{\mathbf{S}}_t = \tau(\mathbf{S}_t, k)$ // Thresholding
    $\mathbf{U}_t = \begin{cases} \text{QR}(\hat{\mathbf{S}}_t) & \text{// Promotes orthogonality} \\ \hat{\mathbf{S}}_t / \|\hat{\mathbf{S}}_t\|_2 & \text{// Promotes sparsity} \end{cases}$
    $\mathbf{E}_t = \mathbf{U}_{t-1}^\top \mathbf{U}_t$
end for
OUTPUT: $\mathbf{U}_t \in \mathbb{R}^{n \times r}$

// THRESHOLDING $\hat{\mathbf{S}}_t = \tau(\mathbf{S}_t, k)$

PROCEDURE
for $i = 1, 2, \ldots, r$ do
    $\mathbf{s}_i = \mathbf{S}_t(:, i)$
    Find the set $\mathcal{T}_t \subset [1, 2, \ldots, n]$ containing indices of k strongest elements of $\mathbf{s}_i$
    Form $\hat{\mathbf{S}}_t(:, i) = \hat{\mathbf{s}}_i$, where $\hat{\mathbf{s}}_i(j) = \begin{cases} \mathbf{s}_i(j) & \text{if } j \in \mathcal{T}_t \\ 0 & \text{if } j \notin \mathcal{T}_t \end{cases}$
end for
OUTPUT: $\hat{\mathbf{S}}_t \in \mathbb{R}^{n \times r}$

Algorithm 4: OPIT - Online Power Iteration via Thresholding

$\mathbf{U}_t$ is then derived from spectral analysis of $\mathbf{R}_t$ which is updated continuously by

$$\mathbf{R}_t = \beta \mathbf{R}_{t-1} + \mathbf{x}_t \mathbf{x}_t^\top. \tag{4.6}$$

Together with the fact that $\text{QR}(\mathbf{R}_t \mathbf{U}_{t-1}) = \text{QR}(\mathbf{C}_t \mathbf{U}_{t-1})$, we can rewrite the first step of PI as follows

$$\mathbf{S}_t = \mathbf{R}_t \mathbf{U}_{t-1} = \beta \mathbf{R}_{t-1} \mathbf{U}_{t-1} + \mathbf{x}_t \mathbf{z}_t^\top, \tag{4.7}$$

where $\mathbf{z}_t = \mathbf{U}_{t-1}^\top \mathbf{x}_t$.

Towards a fast subspace estimator, we can utilize the previous subspace as a warm start in the tracking process. Hereby, a key step at each time t is

to project $\mathbf{U}_t$ into the column space of $\mathbf{U}_{t-1}$, i.e.,

$$\mathbf{U}_t = \mathbf{U}_{t-1}\mathbf{E}_t + \mathbf{U}_{t-1,\perp}\mathbf{F}_t, \tag{4.8}$$

where $\mathbf{U}_{t-1,\perp}$ is the orthogonal complement of $\mathbf{U}_{t-1}$, $\mathbf{E}_t = \mathbf{U}_{t-1}^\top\mathbf{U}_t$ and $\mathbf{F}_t = \mathbf{U}_{t-1,\perp}^\top\mathbf{U}_t$ are coefficient matrices. Specifically, the first term of (4.8) represents the "old" information in $\mathbf{U}_t$, while the second one is its distinctive new information. Substituting $\mathbf{U}_{t-1}$ according to (4.8) (one time-step delayed) into (4.7) results in

$$\mathbf{S}_t = \beta\mathbf{S}_{t-1}\mathbf{E}_{t-1} + \beta\mathbf{R}_{t-1}\mathbf{U}_{t-2,\perp}\mathbf{F}_{t-1} + \mathbf{x}_t\mathbf{z}_t^\top. \tag{4.9}$$

The complement of projecting $\mathbf{x}_t$ into the subspace $\mathbf{U}_{t-1}$ at time t can be given by

$$\mathbf{y}_t = \left(\mathbf{I} - \mathbf{U}_{t-1}\mathbf{U}_{t-1}^\top\right)\mathbf{x}_t = \mathbf{x}_t - \mathbf{U}_{t-1}\mathbf{z}_t. \tag{4.10}$$

Here, $\mathbf{y}_t$ is orthogonal to the column space of $\mathbf{U}_{t-1}$. For short, we denote $\Delta\mathbf{U}_{t-1} = \mathbf{U}_{t-2,\perp}\mathbf{F}_{t-1}$. Based on (4.10), we obtain another expression of $\Delta\mathbf{U}_{t-1}$ as follows

$$\Delta\mathbf{U}_{t-1} = \mathbf{y}_{t-1}\mathbf{h}_{t-1}^\top \quad \text{where} \quad \mathbf{h}_{t-1} = \mathbf{U}_{t-1}^\top\mathbf{y}_{t-1}. \tag{4.11}$$

Under the assumption that the underlying subspace is fixed or slowly varying with time (i.e., $\mathbf{U}_{t-2}\mathbf{U}_{t-2}^\top \simeq \mathbf{U}_{t-1}\mathbf{U}_{t-1}^\top$), $\mathbf{y}_{t-1}$ is nearly orthogonal to the subspace $\mathbf{U}_{t-1}$. In other words, angles between $\mathbf{y}_{t-1}$ and columns of $\mathbf{U}_{t-1}$ are very close to $\pi/2$, and hence, the norm of $\mathbf{h}_{t-1}$ in (4.11) is very small. Therefore, $\Delta\mathbf{U}_{t-1}$ and $\mathbf{R}_{t-1}\Delta\mathbf{U}_{t-1}$ are negligible and can be ignored during the tracking process without any major performance degradation. It stems from the fact that the presence of a small perturbation does not really affect the performance of power methods [167]. Accordingly, a good approximation to (4.9) can be given by

$$\mathbf{S}_t \simeq \beta\mathbf{S}_{t-1}\mathbf{E}_{t-1} + \mathbf{x}_t\mathbf{z}_t^\top. \tag{4.12}$$

In this work, the update (4.12) is further followed by an appropriate perturbation $\mathbf{G}_t$ defined by the following thresholding operation $\tau(.)$ as:

$$\hat{\mathbf{S}}_t \triangleq \tau(\mathbf{S}_t, k) = \mathbf{C}_t\mathbf{U}_{t-1} + \mathbf{G}_t, \tag{4.13}$$

where the thresholding factor k can be determined as in Algorithm 4. Here, $\hat{\mathbf{S}}_t$ is particularly derived from $\mathbf{S}_t$ by keeping the k strongest (absolute value) elements in each column of $\mathbf{S}_t$ and setting the remaining elements to zero. Then, the second step of PI is replaced with

$$\mathbf{U}_t = \begin{cases} \text{QR}(\hat{\mathbf{S}}_t) & \text{if orthonormalization,} \\ \hat{\mathbf{S}}_t / \|\hat{\mathbf{S}}_t\|_2 & \text{if normalization.} \end{cases} \tag{4.14}$$

In addition to the nice property (4.5), another main motivation for using the thresholding operation $\tau(.)$ stems from the following proposition:

Proposition 9 *Denote by $\{\lambda_i\}_{i=1}^n$ the set of singular values of $\mathbf{C}_t$ in descending order (i.e. $\lambda_i \geq \lambda_{i+1}$). When the perturbation $\mathbf{G}_t$ satisfies: $\|\mathbf{G}_t\|_2 \leq \xi(\lambda_r - \lambda_{r+1})$ and $\|\mathbf{A}_t^\top \mathbf{G}_t\|_2 \leq \xi(\lambda_r - \lambda_{r+1}) \cos\theta(\mathbf{A}_t, \mathbf{U}_{t-1})$ for some $\xi < 1$, we obtain*

$$\tan\theta\big(\mathbf{A}_t, \mathbf{C}_t\mathbf{U}_{t-1} + \mathbf{G}_t\big) \leq \gamma \tan\theta\big(\mathbf{A}_t, \mathbf{U}_{t-1}\big),$$

where $0 < \gamma < 1$ and $\theta(.,.)$ denotes the canonical angle (the largest principal angle) between two subspaces.

Proof. Its proof follows immediately Lemma 2.2 in [167].

As a corollary, the estimated $\mathbf{U}_t$ will get closer to the true subspace $\mathbf{A}_t$ with time.

The OPIT algorithm introduces the window parameter W. Here, the inclusion of W is useful in some applications where we often collect multiple data samples instead of a single sample at each time t. The main steps of OPIT are summarized in Algorithm 4.

Complexity: For convenience of analysis, we suppose the window length $W = 1$. Most of the steps in OPIT require a computational complexity of $O(nr^2)$ except the thresholding operator which costs $O(nr + rk\log k)$ operations. Thus, the overall computational complexity of OPIT is $O(\max\{nr, k\log k\}r)$. In terms of memory storage, OPIT does not need to go back past observations but utilizes their information in a recursive way. Hence, the proposed algorithm requires a space of nr elements for saving the estimate $\mathbf{U}_t$, while two buffer matrices $\mathbf{S}_t$ and $\mathbf{E}_t$ need only $nr + r^2$ elements in total. In conclusion, the space complexity of OPIT is linear to the data dimension n.

4.3.2 OPIT with Deflation

A low cost subspace tracking algorithm with linear complexity of computation $O(nr)$ is always preferable due to its fast implementation time, especially for real-time applications.[3] Here, we derive a fast variant of OPIT using deflation called OPITd which can achieve such a complexity while preserving the algorithm's accuracy in most cases.

Our main motivation stems from the fact that if we apply the following

---

[3]With respect to computational complexity, subspace tracking algorithms are categorized into three groups: high complexity $O(n^2r)$ and $O(n^2)$, moderate complexity $O(nr^2)$, and low complexity $O(nr)$. The last group, which is referred to as fast algorithms, is the most important class for online processing [20].

INPUT: $\{\mathbf{x}_i\}_{i=1}^{T}, \mathbf{x}_i \in \mathbb{R}^{n \times 1}$, target rank r, a forgetting factor $0 < \beta \leq 1$, and a thresholding factor k

$$k = \begin{cases} \lfloor (1 - \omega_{sparse})n \rceil & \text{if } \omega_{sparse} \text{ is given,} \\ \lfloor 10r \log n \rceil & \text{if } \omega_{sparse} \text{ is unknown,} \end{cases}$$

where ω_{sparse} is the sparsity level of the sparse basis.

INITIALIZATION: $\mathbf{U}_0 = \text{randn}(n, r)$, $\mathbf{S}_0 = \mathbf{0}_{n \times r}$, $\mathbf{e}_0 = \mathbf{1}_{r \times 1}$.
// Denote $\mathbf{u}_{t,j} = \mathbf{U}_t(:, j)$, $\mathbf{s}_{t,j} = \mathbf{S}_t(:, j)$, and $e_{t,j} = \mathbf{e}_t(j)$.

MAIN PROGRAM:

for $t = 1, 2, \ldots, T$ **do**
    for $j = 1, 2, \ldots, r$ **do**
        $z_{t,j} = \mathbf{u}_{t-1,j}^{\top} \mathbf{x}_t$
        $\mathbf{s}_{t,j} = \beta \frac{t-1}{t} e_{t-1,j} \mathbf{s}_{t-1,j} + \frac{1}{t} z_{t,j} \mathbf{x}_t$
        $\hat{\mathbf{s}}_{t,j} = \tau(\mathbf{s}_{t,j}, k)$ // Thresholding
        $\mathbf{u}_{t,j} = \hat{\mathbf{s}}_{t,j} / \|\hat{\mathbf{s}}_{t,j}\|_2$
        $e_{t,j} = \mathbf{u}_{t-1,j}^{\top} \mathbf{u}_{t,j}$
        $\mathbf{x}_t = \mathbf{x}_t - z_{t,j} \mathbf{u}_{t,j}$ // Deflation
    end for
end for
OUTPUT: $\mathbf{U}_t \in \mathbb{R}^{n \times r}$

Algorithm 5: OPITd - OPIT with Deflation

projection deflation

$$\widetilde{\mathbf{C}}_t = (\mathbf{I} - \mathbf{u}_1 \mathbf{u}_1^{\top}) \mathbf{C}_t (\mathbf{I} - \mathbf{u}_1 \mathbf{u}_1^{\top}), \tag{4.15}$$

where $\mathbf{u}_1$ is the most dominant eigenvector of $\mathbf{C}_t$, then the eigenvectors of $\widetilde{\mathbf{C}}_t$ are exactly the same as $\mathbf{C}_t$ with eigenvalues $\{0, \lambda_2, \ldots, \lambda_n\}$. Here, λ_i is the i-th strongest eigenvalue of $\mathbf{C}_t$. It demonstrates that the deflation (4.15) can eliminate the influence of $\mathbf{u}_1$ (i.e., by setting λ_1 to zero) and switches the second dominant eigenvector up. As a result, once we estimated $\mathbf{u}_1$ by using a specific (online) method, the second dominant eigenvector of $\mathbf{C}_t$ can be extracted from $\widetilde{\mathbf{C}}_t$ in the same way as to $\mathbf{u}_1$. Moreover, repeating this procedure r times can result in r leading eigenvectors of $\mathbf{C}_t$. Interestingly, in the case even when $\mathbf{u}_1$ is not a true eigenvector of $\mathbf{C}_t$, the projection deflation (4.15) still retains desirable properties (e.g. positive semi-definiteness) that may be lost to other deflation transformations [168]. In what follows, we describe the way how to linearize the production of OPIT using the projection deflation (4.15).

To update the j-th column $\mathbf{u}_{t,j}$ of $\mathbf{U}_t$, for $j = 1, 2, \ldots, r$, we replace the

recursive rule (4.12) with

$$\mathbf{s}_{t,j} = \beta \frac{t-1}{t} e_{t-1,j} \mathbf{s}_{t-1,j} + \frac{1}{t} z_{t,j} \mathbf{x}_t, \quad \text{with} \tag{4.16a}$$

$$z_{t,j} = \mathbf{u}_{t-1,j}^\top \mathbf{x}_t \quad \text{and} \quad e_{t-1,j} = \mathbf{u}_{t-2,j}^\top \mathbf{u}_{t-1,j}, \tag{4.16b}$$

where $\mathbf{s}_{t,j}$, $z_{t,j}$, and $e_{t-1,j}$ play the same role as $\mathbf{S}_t$, $\mathbf{z}_t$, and $\mathbf{E}_t$ in (4.12), respectively. Next, the thresholding operation (4.13) boils down to

$$\hat{\mathbf{s}}_{t,j} = \tau(\mathbf{s}_{t,j}, k). \tag{4.17}$$

Then, the column $\mathbf{u}_{t,j}$ is simply derived from normalizing (4.17) to unit length as $\mathbf{u}_{t,j} = \hat{\mathbf{s}}_{t,j}/\|\hat{\mathbf{s}}_{t,j}\|_2$. At the end of the column-wise update, we deflate the component $\mathbf{u}_{t,j}$ from $\mathbf{x}_t$ as $\mathbf{x}_t \leftarrow \mathbf{x}_t - z_{t,j}\mathbf{u}_{t,j}$ for the estimation of the next component $\mathbf{u}_{t,j+1}$. The main steps of OPITd are summarized in Algorithm 5.

Complexity: The most expensive computation comes from the thresholding operation $\tau(\mathbf{s}_{t,j}, k)$ which requires a cost of $O(n+k\log k)$. The remaining steps of OPITd require a computational complexity of $O(n)$ only. Accordingly, OPITd costs a complexity of $O(r \max\{n, k\log k\})$ for updating the whole matrix $\mathbf{U}_t$ at each time t. In practice, we often set the value of k to $O(r\log n)$ or $\lfloor (1-\omega_{sparse})n \rceil$ which is much smaller than n, and thus, the overall complexity of OPITd is approximately linear to nr. OPITd also requires a less memory storage than OPIT. Specifically, its space complexity is $2nr + r$ for saving $\mathbf{U}_t$, $\mathbf{S}_t = [\mathbf{s}_{t,1}, \mathbf{s}_{t,2}, \dots, \mathbf{s}_{t,r}]$ of size $n \times r$ and $\mathbf{e}_t = [e_{t,1}, e_{t,2}, \dots, e_{t,r}]^\top$ of size $r \times 1$ at time t.

4.3.3   Discussions

First, it is worth noting that both OPIT and OPITd cannot enforce orthogonality and sparsity in the estimate at the same time. On the one hand, when we adopt the orthonormalization step using the QR factorization, OPIT ensures orthogonality but lacks sparsity. Although performing the QR step can increase the numerical stability of OPIT, it destroys the sparsity, especially when the target rank r is not too small. In most cases, the Q-factor of the thresholded $\hat{\mathbf{S}}_t$ is a dense (orthogonal) matrix. However, when the columns of $\hat{\mathbf{S}}_t$ are sufficiently sparse and have mostly non-zero elements in non-overlapping sets in its row support, then $\hat{\mathbf{S}}_t$ is almost orthogonal and its Q-factor can be nearly sparse. We particularly meet such a case when data streams are high-dimensional but of very low rank (i.e., $r \ll n$) and/or the sparsity level ω_{sparse} is extremely high. In fact, we often emphasize the principal subspace rather than its specific basis in subspace tracking, thus the lack of sparsity of OPIT is not the issue. On the other hand, when the normalization step (e.g. $\mathbf{U}_t = \hat{\mathbf{S}}_t/\|\hat{\mathbf{S}}_t\|_2$) is taken into account instead of the QR step,

OPIT results in a sparse but non-orthogonal mixing matrix $\mathbf{U}_t$. The operation requires only $O(nr)$ while the QR step costs a complexity of $O(nr^2)$. Therefore, it helps speed up the computation of OPIT especially when r is reasonably high compared to the dimension n. More importantly, with this simple normalization, OPIT can achieve excellent subspace estimation accuracy against the state-of-the-art SST algorithms, please see Figs. 4.4 and 4.5 for examples.

OPITd promotes sparsity but entails non-orthogonality and sub-optimality. Thanks to the projection deflation, OPITd offers a fast column-wise update for tracking the underlying subspace and successes in achieving the sparsity. The deflation has the advantage to estimate the eigenvectors (which is referred to as principal components) while the matrix $\mathbf{U}_t$ in OPIT can be any basis of the principal subspace (not necessarily the eigenvectors). Accordingly, OPITd has benefits in some applications such as data whitening requiring the eigenvectors. Specifically, the combination of the thresholding operation $\tau(\mathbf{s}_{t,j}, k)$ and the column normalization results directly in sparse components in the estimate $\mathbf{U}_t$ at each time t. However, the deflation may cause loss of orthogonality and introduces cumulative errors which can affect the successive estimation of the next component. Accordingly, when the target rank r is not too small compared to the data dimension n, both convergence rate and estimation accuracy of OPITd are less than that of OPIT, see Fig. 4.7(b) for an illustration. In such a situation, we can re-orthonormalize $\mathbf{U}_t$ after a period of time to remedy the issue at low cost as well as increase the numerical stability of OPITd.

Next, how to choose the value of k? Ideally, this factor must be a $r \times 1$ vector $[k_1, k_2, \ldots, k_r]$ where k_j represents the threshold level for the j-th column $\mathbf{A}_t(:, j)$. Clearly, the value of k_j should be close to the number of non-zero elements in $\mathbf{A}_t(:, j)$. Without loss of generality, we can assume that sparse patterns in $\mathbf{A}_t$ are uniformly distributed, i.e., $k_i \simeq k_j \; \forall i, j$. Accordingly, we can set $k \simeq k_j \simeq \lfloor (1 - \omega_{sparse})n \rceil$ when the prior knowledge of the sparsity level ω_{sparse} – the percentage of non-zero elements in $\mathbf{A}_t$ – is given. If this information is not available, we can tune this factor through cross-validation or simply chosen in $O(\log n)$, e.g. $k = \lfloor mr \log n \rceil$ where m is a positive number. The former remedy is useful for batch sparse subspace estimation and sparse PCA [169]. However, it requires a validation set – which we have to pass a number of observations several times – and hence turns out to be inefficient for tracking problems. The latter one is very simple and capable of achieving reasonable performance in practice. It stems from the rigorous evidence in [170–172] that sparse subspace/PCA algorithms can recover the sparse principal components in polynomial time when the expected number of non-zero elements in each component is at most $O(\sqrt{T/\log n})$. As indicated later in Section IV, the number of observations $T = O(n)$ can guarantee OPIT's convergence, please see the condition (4.18). Furthermore, we have

$\log n < \sqrt{T/\log n}$ when $T = O(n)$ for a large n, and thus, we can choose the factor k in the logarithmic regime $O(\log n)$ to ensure the thresholded matrix is sufficiently sparse. A natural question raised here is whether the tracking ability of OPIT deteriorates or not when the number of selected elements is smaller than the actual number of non-zeros in $\mathbf{A}_t$? (e.g. it might occur due to the low level of sparsity). Fortunately, Proposition 9 also suggests that if the perturbation error caused by the choice of k is small enough, OPIT still results in a good estimate of $\mathbf{A}_t$ when the number of observation is large enough.

Compared to the state-of-the-art power-based subspace tracking algorithms, OPIT is more elegant, refined, and effective. Particularly upon the arrival of new data $\mathbf{x}_t$, many power-based subspace trackers (e.g., Oja-types, NP-types, and DPM-types) adopt the update rule $\mathbf{U}_t = \text{orthnorm}(\mathbf{U}_{t-1} + \eta_t \mathbf{x}_t \mathbf{z}_t^\top)$ where η_t is the step size and orthnorm(.) is an orthonormalization procedure [20]. Therefore, the inclusion of $\mathbf{E}_{t-1}$ in (4.12) not only makes OPIT different from them, but also greatly bolsters its tracking ability. The matrix $\mathbf{E}_{t-1}$, which contains cosines of the principal angles between two successive subspaces, plays the role of feedback in the tracking process. Accordingly, it could help improve the adaptation rate and stability of OPIT, especially in nonstationary environments. API-type subspace trackers, on the other hand, exploit the projection approximation $\mathbf{U}_t \simeq \mathbf{U}_{t-1} \Theta_t$ where Θ_t is nearly orthogonal and very close to an identity matrix [154]. Hereby, they would predict the current tracking performance error and then use it for estimating the true subspace. More specifically, they follow the update rule $\mathbf{U}_t = \mathbf{U}_{t-1} \Theta_t + \mathbf{y}_t \mathbf{g}_t^\top \Theta_t$ where $\mathbf{y}_t$ is the complement (error) of projecting $\mathbf{x}_t$ onto $\mathbf{U}_{t-1}$ defined as in (4.10), $\mathbf{g}_t$ is a gain vector, and $\Theta_t = (\mathbf{I}_r + \|\mathbf{y}_t\|^2 \mathbf{g}_t \mathbf{g}_t^\top)^{-1/2}$. However, when abrupt changes happen (e.g., due to impulsive noises and outliers or data drift), the error $\mathbf{y}_t$ would be very large. The state transition matrix Θ_t would be very far from ideal that could degrade their subspace estimation accuracy as well as convergence rate, see Section E.1 in our supplementary document for examples. By contrast, OPIT exploits the past tracking performance error (i.e., one time step delayed) caused by itself which is independent of the current error $\mathbf{y}_t$. Thus, OPIT is less sensitive to such changes than API-types. Together with the hard-thresholding operator $\tau(.)$ in (4.13), OPIT stands out from all the rest. The tracking ability of OPIT is verified by several experiments in Section V where the results indicate that OPIT outperforms completely the-state-of-the-art subspace trackers (including several power-based methods) in both classical and high dimension regimes.

4.4 Convergence Analysis

In this section, we provide a convergence analysis for the proposed OPIT algorithm in Algorithm 4 under the assumption that $\mathbf{A}_t = \mathbf{A}$ is unchanged over time and $\beta = 1$.[4]

We make the following assumptions to facilitate our convergence analysis:

(A1) $\mathbf{A}$ is chosen in the set $\mathcal{U} = \{\mathbf{U} \in \mathbb{G}_{n,r}, \|\mathbf{U}\|_{*,0} \leq (1 - \omega_{sparse})n$, and $\|\mathbf{U}\|_2 = 1\}$, where $\mathbb{G}_{n,r}$ denotes the class of $n \times r$ well-condition matrices and $\|\mathbf{U}\|_{*,0} = \max_j \|\mathbf{U}(:,j)\|_0$. Here, the parameter ω_{sparse} represents the sparsity level of $\mathbf{A}$. In addition, $\mathbf{A}$ is sparse enough in the sense that the average number of non-zero elements in each column is at most $\sqrt{n/\log n}$.

(A2) Data samples $\{\mathbf{x}_t\}_{t \geq 1}$ are norm-bounded, i.e., $\|\mathbf{x}_t\|_2 \leq M < \infty \ \forall t$. Low-rank signals $\{\boldsymbol{\ell}_t\}_{t \geq t}$ are supposed to be deterministic and bounded. Noise vectors $\{\mathbf{n}_t\}_{t \geq 1}$ are i.i.d. random variables of zero mean and their power is lower than the signal power.

In (A1), the underlying subspace is supposed to be sparse in the sense of *column sparsity* defined by Vu *et al.* in [173].[5] It is not a strict sparsity constraint as the set $\mathcal{U}$ covers several supports such as row-sparse, elementwise-sparse, and local region-sparse. Besides, the unit-norm constraint of (A1) is a very mild condition as we can rescale $\mathbf{A}$ by recasting its operator norm into the signal power. The second constraint of (A1) ensures trackers to estimate the sparse subspace with high probability [170]. Meanwhile, (A2) is a common assumption for subspace tracking problems and holds in many situations [25]. Together with (A1), they help prevent the ill-conditioned computation and support the perturbation analysis of QR decomposition due to the thresholding operation.

Given these assumptions, the main theoretical result of OPIT's convergence can be stated by the following theorem:

---

[4]We limit our analysis in this work to a stationary case when $\mathbf{A}_t = \mathbf{A} \ \forall t$ and $\beta = 1$. Establishing the ϵ-relative-error approximation guarantee for OPIT in nonstationary environments is non-trivial as data samples do not share the same population. Specifically, finding a tight upper bound on the error matrix $\Delta \mathbf{C}_t$ – which plays a key role in establishing the two necessary conditions (4.18) and (4.19) as well as Lemmas 1 and 2 – is challenging. Instead of the normal sample covariance matrix (SCM), an exponential weighted variant of the SCM is applied here because of the forgetting factor $\beta < 1$. It would make the theoretical convergence analysis more complicated. We leave this challenge for future work.

[5]With respect to the concept of subspace sparsity, Vu *et al.* in [173] introduced two notions: *column sparsity* and *row sparsity*. Specifically, a subspace is said to be column sparse if some orthonormal basis contains sparse vectors. Meanwhile, every orthonormal basis of a row sparse subspace must consist of sparse vectors. Accordingly, row sparse subspaces also belong to the class of column sparse subspaces. In this work, the proposed OPIT algorithm can achieve an ϵ-relative-error approximation guarantee for the class of column sparse subspaces, and thus, its convergence guarantee also holds under the row sparsity.

Theorem 3 *Suppose that* $\mathbf{A}_t = \mathbf{A}$, $\beta = 1$, *the true covariance matrix has the form* $\mathbf{C} = \sigma_x^2 \mathbf{A}\mathbf{A}^\top + \sigma_n^2 \mathbf{I}$, *and two assumptions (A1)-(A2) are met. The initialization matrix* $\mathbf{U}_0$ *and the number of observed (block) data samples t satisfies the following conditions*

$$t \geq \frac{C \log(2/\delta)}{W \epsilon^2} \left(\sqrt{r} + \left(\frac{\sigma_n^2}{\sigma_x^2} + 2\frac{\sigma_n}{\sigma_x} \right) \sqrt{n} \right)^2, \qquad (4.18)$$

$$\max \left\{ \sin \theta(\mathbf{A}, \mathbf{U}_0), \epsilon \right\} \leq \left(\frac{3 - 2\sqrt{2}}{r + 2\sqrt{r}(\sqrt{2} - 1)} \right)^{1/2}, \qquad (4.19)$$

where $\epsilon > 0$ *is a predefined accuracy, C is a universal positive number and* $0 < \delta \ll 1$ *is a predefined error probability. At time t, when* $\mathbf{U}_t$ *is generated by OPIT with the orthonormalization step using QR factorization, then*

$$d_t \overset{\Delta}{=} \sin \theta(\mathbf{A}, \mathbf{U}_t) \leq \epsilon, \qquad (4.20)$$

with a probability at least $1 - \delta$.

Proof Sketch. First, let us denote the QR decomposition of $\mathbf{S}_t$ by $\mathbf{S}_t = \mathbf{U}_{t,\mathcal{F}}\mathbf{R}_{t,\mathcal{F}}$ where "$\mathcal{F}$" stands for "full" entries. Here, we can express $\mathbf{U}_t = \mathbf{U}_{t,\mathcal{F}}\mathbf{W}_1 + \mathbf{U}_{t,\mathcal{F},\perp}\mathbf{W}_2$ where $\mathbf{U}_{t,\mathcal{F},\perp} \in \mathbb{R}^{n \times (n-r)}$ is the orthogonal complement of $\mathbf{U}_{t,\mathcal{F}}$(i.e., $\mathbf{U}_{t,\mathcal{F}}^\top \mathbf{U}_{t,\mathcal{F},\perp} = \mathbf{0}$), $\mathbf{W}_1 \in \mathbb{R}^{r \times r}$ and $\mathbf{W}_2 \in \mathbb{R}^{(n-r) \times r}$ are coefficient matrices. Specifically, it is easy to obtain that $\|\mathbf{W}_1\|_2 = \|\mathbf{U}_{t,\mathcal{F}}^\top \mathbf{U}_t\|_2$ and $\|\mathbf{W}_2\|_2 = \|\mathbf{U}_{t,\mathcal{F},\perp}^\top \mathbf{U}_t\|_2$. Accordingly, we can bound the distance $d_t = \sin \theta(\mathbf{A}, \mathbf{U}_t)$ as follows:[6]

$$
\begin{aligned}
d_t &= \left\| \mathbf{A}_\perp^\top \mathbf{U}_t \right\|_2 = \left\| \mathbf{A}_\perp^\top \left(\mathbf{U}_{t,\mathcal{F}}\mathbf{W}_1 + \mathbf{U}_{t,\mathcal{F},\perp}\mathbf{W}_2 \right) \right\|_2 \\
&\overset{(i)}{\leq} \left\| \mathbf{A}_\perp^\top \mathbf{U}_{t,\mathcal{F}} \right\|_2 \left\| \mathbf{W}_1 \right\|_2 + \left\| \mathbf{A}_\perp^\top \mathbf{U}_{t,\mathcal{F},\perp} \right\|_2 \left\| \mathbf{W}_2 \right\|_2 \overset{(ii)}{\leq} \left\| \mathbf{A}_\perp^\top \mathbf{U}_{t,\mathcal{F}} \right\|_2 + \left\| \mathbf{U}_{t,\perp}^\top \mathbf{U}_{t,\mathcal{F}} \right\|_2.
\end{aligned}
$$
$$(4.21)$$

Here, (i) thanks to the standard inequalities $\|\mathbf{M} + \mathbf{N}\|_2 \leq \|\mathbf{M}\|_2 + \|\mathbf{N}\|_2$ and $\|\mathbf{M}\mathbf{N}\|_2 \leq \|\mathbf{M}\|_2 \|\mathbf{N}\|_2$; and (ii) is due to the following facts: $\|\mathbf{A}_\perp\|_2 = \|\mathbf{U}_t\|_2 = \|\mathbf{U}_{t,\mathcal{F},\perp}\|_2 = 1$, $\|\mathbf{W}_1\|_2 \leq \|\mathbf{U}_{t,\mathcal{F}}^\top\|_2 \|\mathbf{U}_t\|_2 \leq 1$, $\|\mathbf{A}_\perp^\top \mathbf{U}_{t,\mathcal{F},\perp}\|_2 \leq \|\mathbf{A}_\perp^\top\|_2 \|\mathbf{U}_{t,\mathcal{F},\perp}\|_2 \leq 1$, and $\|\mathbf{U}_{t,\mathcal{F},\perp}^\top \mathbf{U}_t\|_2 = \|\mathbf{U}_{t,\perp}^\top \mathbf{U}_{t,\mathcal{F}}\|_2$.

The two terms of the right hand side of (4.21) can be bounded by Lemma 5 and 6, respectively.

---

[6]For any two orthonormal matrices $\mathbf{A}$ and $\mathbf{U}$ of the same size, we always have $\sin \theta(\mathbf{A}, \mathbf{U}) = \|\mathbf{A}_\perp^\top \mathbf{U}\|_2 = \|\mathbf{U}_\perp^\top \mathbf{A}\|_2$.

Lemma 5 *Let $\Delta C_t = C_t - C$, we always have*

$$\left\|A_\perp^\top U_{t,\mathcal{F}}\right\|_2 \leq \frac{\sigma_n^2\left\|A_\perp^\top U_{t-1}\right\|_2 + \left\|\Delta C_t\right\|_2}{\left(\left[\left(\sigma_x^2 + \sigma_n^2\right)\sqrt{1 - \left\|A_\perp^\top U_{t-1}\right\|_2^2} - \left\|\Delta C_t\right\|_2\right]^2 \right.} \tag{4.22}$$
$$\left. + \left[\sigma_n^2\left\|A_\perp^\top U_{t-1}\right\|_2 + \left\|\Delta C_t\right\|_2\right]^2\right)^{1/2}$$

Proof. See Appendix A.

Lemma 6 *The distance between U_t and $U_{t,\mathcal{F}}$ is bounded by*

$$\left\|U_{t,\perp}^\top U_{t,\mathcal{F}}\right\|_2 \leq \frac{\sqrt{r}\left(\sigma_n^2\left\|A_\perp^\top U_{t-1}\right\|_2 + \left\|\Delta C_t\right\|_2\right)}{\left(\left(\sigma_x^2 + \sigma_n^2\right)\sqrt{1 - \left\|A_\perp^\top U_{t-1}\right\|_2^2}\right.}$$
$$\left. - \left(1 + \sqrt{r}(1 + \sqrt{2})\right)\left(\sigma_n^2\left\|A_\perp^\top U_{t-1}\right\|_2 + \left\|\Delta C_t\right\|_2\right)\right)$$
$$\tag{4.23}$$

under the following condition

$$\frac{\sigma_n^2\left\|A_\perp^\top U_{t-1}\right\|_2 + \left\|\Delta C_t\right\|_2}{\left(\sigma_x^2 + \sigma_n^2\right)\sqrt{1 - \left\|A_\perp^\top U_{t-1}\right\|_2^2}} \leq \frac{\sqrt{2} - 1}{\sqrt{r} - 1 + \sqrt{2}}. \tag{4.24}$$

Proof. See Appendix B.

Next, Lemma 7 indicates an upper bound on $\|\Delta C_t\|_2$ which plays a crucial role in Lemma 5 and 6 as well as establishing the two conditions (4.18) and (4.19) for the convergence of OPIT.

Lemma 7 *The error matrix ΔC_t is bounded in the operator norm with a probability at least $1 - \delta$:*

$$\left\|\Delta C_t\right\|_2 \leq c_\delta\left(\sigma_x^2\sqrt{\frac{r}{tW}} + \left(2\sigma_n\sigma_x + \sigma_n^2\right)\sqrt{\frac{n}{tW}}\right), \tag{4.25}$$

where $\delta > 0$ is a predefined error probability, and $c_\delta = C\sqrt{\log(2/\delta)}$ with a universal positive number $C > 0$.

Proof. See Appendix C.

Then, the necessary condition (4.24) for Lemma 6 is particularly satisfied

when (4.18) is met and the following inequality holds

$$\max\left\{\sin\theta(\mathbf{A},\mathbf{U}_0),\epsilon\right\} \leq \sqrt{\frac{\alpha(r,\rho)}{1-\alpha(r,\rho)}}, \quad \text{where} \tag{4.26}$$

$$\alpha(r,\rho) = \frac{(3-2\sqrt{2})(\sigma_x^2+\sigma_n^2)^2}{\left(r+2\sqrt{r}(\sqrt{2}-1)+3-2\sqrt{2}\right)\left(\sigma_n^2+r^{-1}\rho\sigma_x^2\right)^2}, \tag{4.27}$$

for any positive number ρ in the range $(0, r]$, please see Appendix D for details. Clearly, (4.19) provides a lower bound on $\sqrt{\alpha(r,\rho)/(1-\alpha(r,\rho))}$.

Accordingly, Lemma 6 is achieved under the two conditions (4.18) and (4.19) while Lemma 1 holds for all t. Now, given Lemma 5, 6, and 7, the distance d_t can be bounded by Lemma 6.

Lemma 8 *Let $d_0 = \sin\theta(\mathbf{A},\mathbf{U}_0)$, $\omega_0 = \max\{d_0,\epsilon\}$, $\gamma > 0$ is any positive number satisfying $\omega_0 \leq \gamma r\sqrt{1-\omega_0^2}$ and $\rho\gamma < 1$. Suppose that $\omega_0 \leq \sqrt{2}/2$, the two conditions (4.19) and (4.18) are met, we obtain*

$$d_t \leq \frac{r\sigma_n^2+\rho\sigma_x^2}{r\xi\sqrt{1-\omega_0^2}}\max\left\{d_{t-1},\epsilon\right\}, \quad \text{where} \tag{4.28}$$

$$\xi = 0.5\max\left\{\left[(1+\gamma^2r^2)\sigma_n^4+(1-\rho\gamma)^2\sigma_x^4+2(1+\gamma^2r^2-\rho\gamma)\sigma_n^2\sigma_x^2\right]^{1/2},\right.$$
$$\left.(\sigma_n^2+\sigma_x^2)(1-\varrho)/\sqrt{r}\right\}, \tag{4.29}$$

with $\varrho = \gamma\left(1+\sqrt{r}(1+\sqrt{2})(r\sigma_n^2+\rho\sigma_x^2)\right)\left(\sigma_n^2+\sigma_x^2\right)^{-1}$. Furthermore, $d_t \leq \epsilon$ also holds when t satisfies the condition (4.18).

Proof. See Appendix D.

4.5 Experiments

In this section, we conduct several experiments on both synthetic and real data to demonstrate the effectiveness and efficiency of OPIT and its variant OPITd. Their performance is evaluated in comparison with state-of-the-art algorithms. Our simulations are implemented using MATLAB on a laptop of Intel core *i7* and 16GB of RAM. Our codes are also available online at `https://github.com/thanhtbt/sst/` to facilitate replicability and reproducibility.

4.5.1 Experiments with Synthetic Data

4.5.1.1 Experiment Setup

Following the formulation in section 4.2, data samples $\{\mathbf{x}_t\}_{t \geq 1}$ are generated at random under the standard model:

$$\mathbf{x}_t = \mathbf{A}_t \mathbf{w}_t + \sigma_n \mathbf{n}_t, \tag{4.30}$$

where $\mathbf{n}_t \in \mathbb{R}^{n \times 1}$ is a noise vector derived from $\mathcal{N}(\mathbf{0}, \mathbf{I}_n)$, $\sigma_n > 0$ is to control the effect of the noise on algorithm's performance, $\mathbf{w}_t \in \mathbb{R}^{r \times 1}$ is an i.i.d. Gaussian random vector of zero-mean and unit-variance to represent the subspace coefficient. The sparse mixing matrix $\mathbf{A}_t \in \mathbb{R}^{n \times r}$ at time t is simulated as

$$\mathbf{A}_t = \Omega \circledast (\mathbf{A}_{t-1} + \varepsilon \mathbf{N}_t), \tag{4.31}$$

where $\circledast$ denotes the Hadamard product, $\Omega \in \mathbb{R}^{n \times r}$ is a Bernoulli random matrix with probability $1 - \omega_{sparse}$, $\mathbf{N}_t$ is a normalized Gaussian white noise matrix, and $\varepsilon > 0$ is the time-varying factor aimed to control the subspace variation with time.

In order to evaluate the subspace estimation performance, we measure the following distance between two subspaces[7]

$$d_t \overset{\Delta}{=} \sin \theta(\mathbf{A}_t, \mathbf{U}_t), \tag{4.32}$$

where $\mathbf{U}_t$ refers to the estimated subspace at time t.

4.5.1.2 Effect of the forgetting factor β

The choice of the forgetting factor β plays an essential role in the tracking ability of OPIT. We investigated its effect by varying its value from 0.1 to 1 and then evaluating the performance of OPIT. Here, the data dimension, the true rank, the number of data samples were set at $n = 50$, $r = 10$, and $T = 1000$, respectively. We fixed the noise factor at $\sigma_n = 10^{-3}$, while two time-varying levels were considered, namely $\varepsilon = 0$ (stationary) and $\varepsilon = 10^{-3}$ (nonstationary). Results are illustrated in Fig. 4.1. In the stationary environment (Fig. 1(a)), we can see that the higher the value of β is, the better the performance OPIT achieves, and $\beta = 1$ offers the best tracking performance. In the time-varying environment (Fig. 1(b)), $0 \ll \beta < 1$ can provide reasonably high subspace estimation accuracy. When β is close to 0, OPIT can track the underlying subspace over time but its accuracy is low. When $\beta = 1$, OPIT's performance degrades as time passes.

---

[7]Given two orthonormal matrices $\mathbf{A}$ and $\mathbf{U}$ of the same size, we always have $\sin \theta(\mathbf{A}, \mathbf{U}) = \|\mathbf{A}_\perp^\top \mathbf{U}\|_2 = \|\mathbf{U}_\perp^\top \mathbf{A}\|_2 = \|\mathbf{A}\mathbf{A}^\top - \mathbf{U}\mathbf{U}^\top\|_2$ where $(.)_\perp$ denotes the orthogonal complement, e.g., $\mathbf{U}^\top \mathbf{U}_\perp = \mathbf{0}$. In MATLAB, this distance can be easily computed by using the command $\mathtt{sin(subspace(\mathbf{A}, \mathbf{U}))}$.

(a) Stationary: $\epsilon = 0$ (b) Nonstationary: $\epsilon = 10^{-3}$
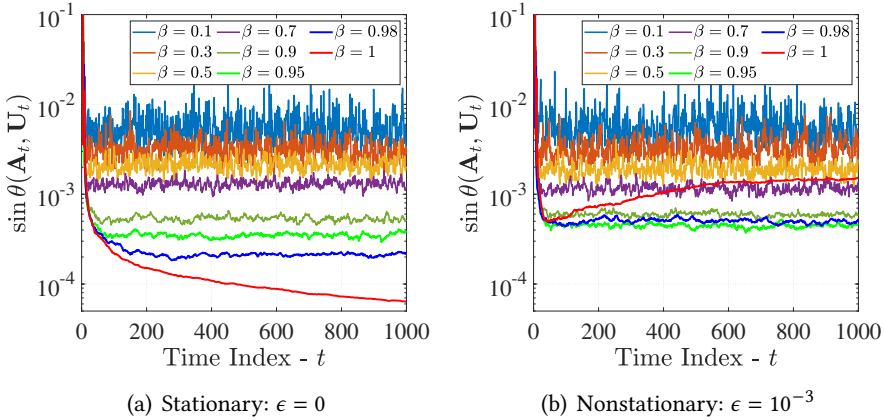
Figure 4.1: Effect of the forgetting factor β.

4.5.1.3 OPIT in Noisy and Dynamic Environments

In order to demonstrate the tracking ability of OPIT in nonstationary environments, we varied the value of the noise level σ_n and the time-varying factor ε among $\{10^{-1}, 10^{-2}, 10^{-3}\}$ and then evaluated its subspace estimation accuracy. Two case studies were considered, including the small-scale $\{n = 100, r = 5\}$ and the large-scale $\{n = 1000, r = 50\}$ in which the sparsity level ω_{sparse} was set to 90% and an abrupt change was created at $t = 500$. The forgetting factor β was fixed at 0.9 in both cases. We set the value of the thresholding factor k to $\lfloor 10r \log n \rfloor$.

Fig. 4.2 and Fig. 7.12 illustrate the effect of the noise level σ_n and the time-varying factor ε on the performance of OPIT, respectively. We can see that the value of σ_n and ε did not affect the convergence rate of OPIT but its estimation error. Despite the value of σ_n and ε, OPIT still tracked successfully the underlying sparse subspace even in the presence of a significant change at $t = 500$. The lower σ_n and ε are, the better subspace estimation accuracy OPIT can achieve. Moreover, these experimental results indicate that the dimension n and rank r had in fact a small impact on how fast OPIT converges in dynamic environments. Specifically, when dealing with the large-scale setting, its convergence rate was faster than that when handling the small-scale one.

4.5.1.4 OPIT versus Other SST Methods

In this task, we compare the performance of OPIT against the state-of-the-art subspace tracking algorithms in different scenarios. These SST algorithms

(a) $n = 100, r = 5$

(b) $n = 1000, r = 50$
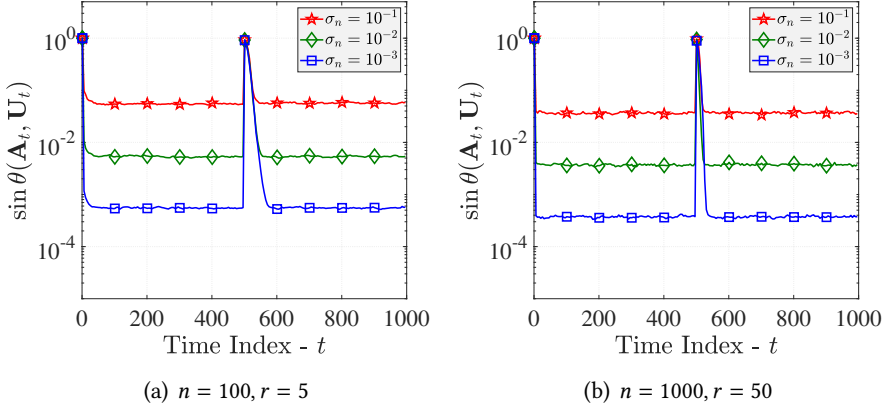
Figure 4.2: Effect of the noise level σ_n on performance of OPIT: sparsity level $\omega_{sparse} = 90\%$, time-varying factor $\varepsilon = 10^{-4}$, and forgetting factor $\beta = 0.9$.

(a) $n = 100, r = 5$

(b) $n = 1000, r = 50$
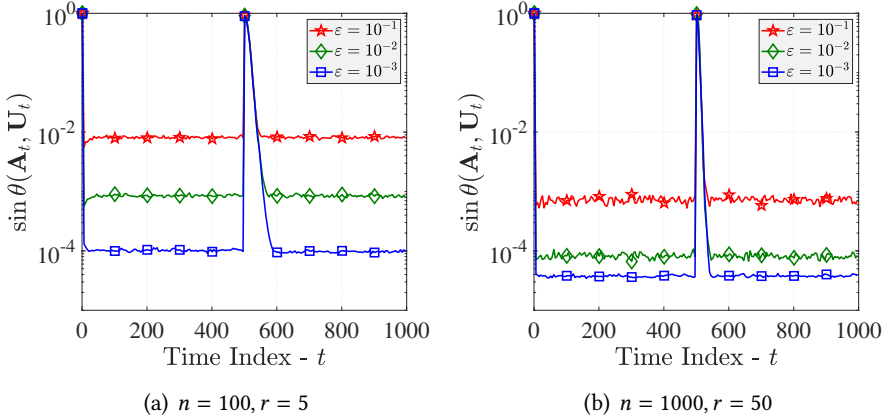
Figure 4.3: Effect of the time-varying factor ε on performance of OPIT: sparsity level $\omega_{sparse} = 90\%$, noise level $\sigma = 10^{-4}$, and forgetting factor $\beta = 0.9$.

include ℓ_1-PAST [95], SS-FAPI [98], SSPCA [94], and AdaOja [149].

We used 1000 snapshots derived from the model (4.30) in which the time-varying factor ε was fixed at 10^{-3} and the value of σ_n was set to two levels: 10^{-1} and 10^{-3}. Here, two sparsity levels were also investigated, including 50% and 90%. The length of window was set to $W = \lfloor \log n \rfloor$ for the large-scale settings and low noise levels, while we used $W = 1$ for others. We fixed the forgetting factor β at 0.97 for all simulations in this task. For OPIT, the normalization step was used instead of the QR factorization. Parameters of

(a) $r = 2$, $\omega_{sparse} = 50\%$

(b) $r = 2$, $\omega_{sparse} = 90\%$

(c) $r = 10$, $\omega_{sparse} = 50\%$

(d) $r = 10$, $\omega_{sparse} = 90\%$
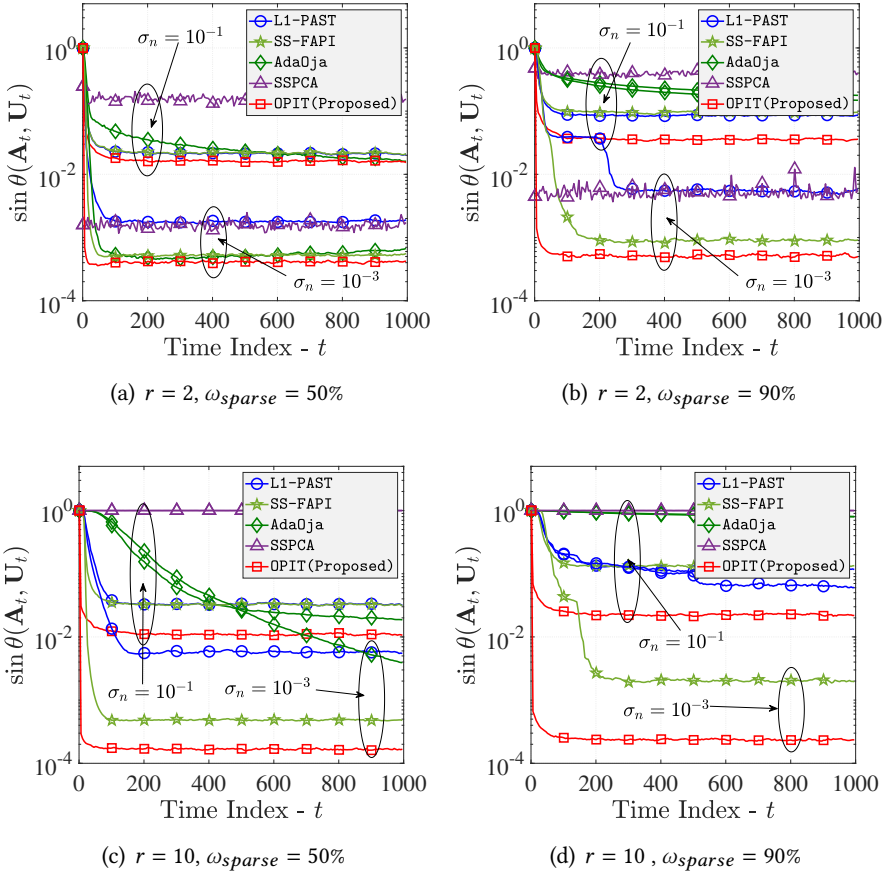
Figure 4.4: Performance comparisons between OPIT and other SST algorithms in the classical setting: dimension $n = 50$, snapshots $T = 1000$, and time-varying factor $\varepsilon = 10^{-3}$.

other SST algorithms were kept default to have a fair comparison.

Experimental results are shown as in Fig. 4.4 and Fig. 4.5. In the classical regime (see Fig. 4.4), OPIT was one of the two best effective SST algorithms, together with SS-FAPI. In particular, the two algorithms outperformed ℓ_1-PAST, SSPCA, and AdaOja in all simulations. Indeed, the convergence rate and estimation accuracy of OPIT were better than than of SS-FAPI, especially in the case of $\omega_{sparse} = 90\%$. When the target rank was set to a very low value ($r = 2$), all SST algorithms were capable of tracking the underlying subspace over time, see Fig. 4.4(a)-(b). When the target rank was reasonably high compared to the dimension ($r = 10$ versus $n = 50$), SSPCA failed while ℓ_1-PAST and AdaOja still worked, but their tracking ability was substantially lower

(a) $n = 1000$, $\omega_{sparse} = 50\%$

(b) $n = 1000$, $\omega_{sparse} = 90\%$

(c) $r = 10000$, $\omega_{sparse} = 50\%$

(d) $n = 10000$, $\omega_{sparse} = 90\%$
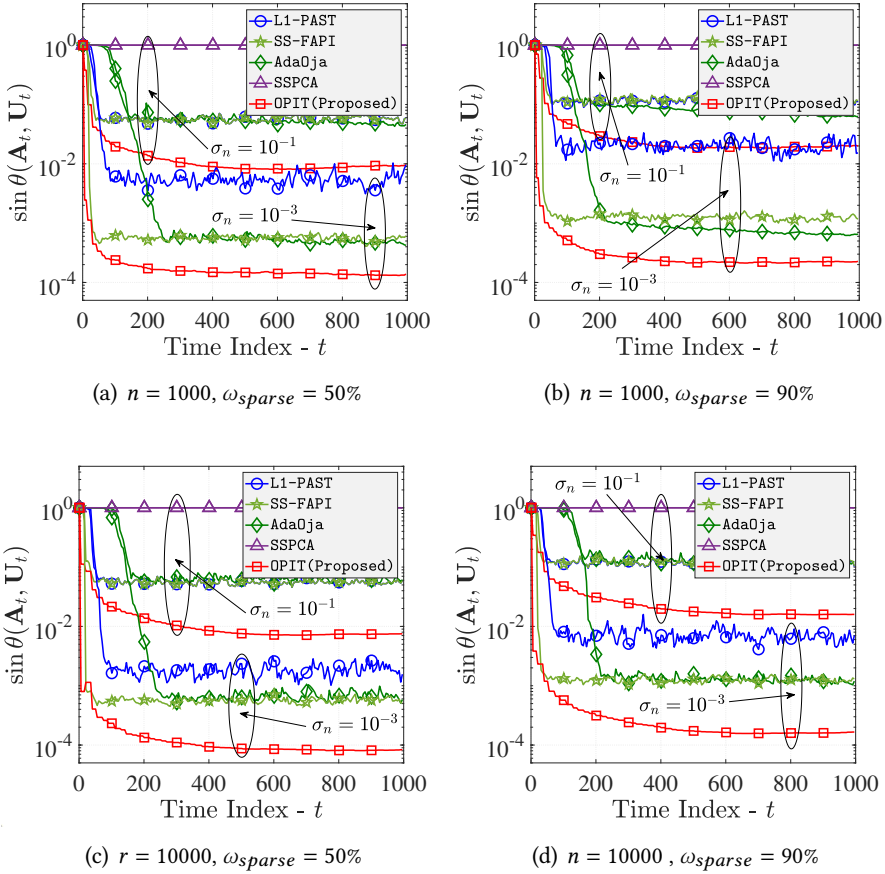
Figure 4.5: Performance comparisons between OPIT and other SST algorithms in high dimensions: target rank $r = 10$, snapshots $T = 1000$, and time-varying factor $\varepsilon = 10^{-3}$.

than SS-FAPI and OPIT, as illustrated in Fig. 4.4(c)-(d).

When dealing with high-dimensional and large-scale settings, OPIT completely outperformed other SST algorithms at both low and high levels of noise as well as sparsity, as shown in Fig. 4.5. SSPCA failed to track the underlying subspace while AdaOja, ℓ_1-PAST, and SS-FAP could work in high dimensions. However, their performance in terms of estimation accuracy and convergence rate were much less than that of OPIT.
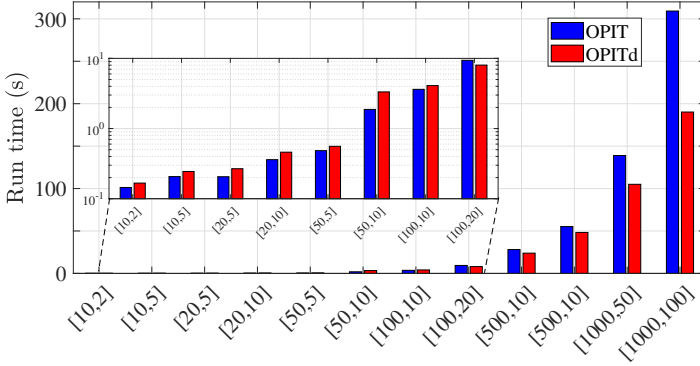
Figure 4.6: OPITd versus OPIT: Run time.

OPITd versus OPIT

We here investigate the tracking ability of OPITd in comparison with the original OPIT with respect to aspects: runtime, estimation accuracy, and robustness to abrupt changes.

To measure how fast OPITd is, we tested many configurations of $\{n, r\}$ and reported its run time. Most other parameters were kept fixed as in the previous task except the number of snapshots T, including the sparsity level $\omega_{sparse} = 90\%$, the noise level $\sigma_n = 10^{-3}$, the time-varying factor $\varepsilon = 10^{-3}$, and the forgetting factor $\beta = 0.97$. We used 3000 snapshots instead of 1000 for this task. The experimental results in Fig. 4.6 show that OPITd was faster than OPIT when the dimension n and the target rank r were set to large values ($n \geq 100$ and $r \geq 10$), especially when the dimension n is actually high, e.g. $n = 1000$.

We next investigate the tracking ability of OPITd in time-varying environments with abrupt changes. We reused the experiment setup above and created two abrupt changes at $t = 1000$ and $t = 2000$ to evaluate how fast OPITd converges. Two noise levels were considered, including $\sigma_n = 10^{-1}$ and $\sigma_n = 10^{-3}$. The results are illustrated in Fig. 4.7 and Fig. 4.8. When the underlying model was of low rank, OPITd had almost the same performance to OPIT, see Fig. 4.7(a). When the target rank r was large, OPITd did not work well, probably because the projection deflation might lead to a cumulative error between successive estimates. However, if the value of r is not too large, OPITd could track successfully the underlying subspace over time when the sparsity level ω_{sparse} was not too high, as shown in Fig. 4.8.

(a) $r = 5$ (b) $r = 30$
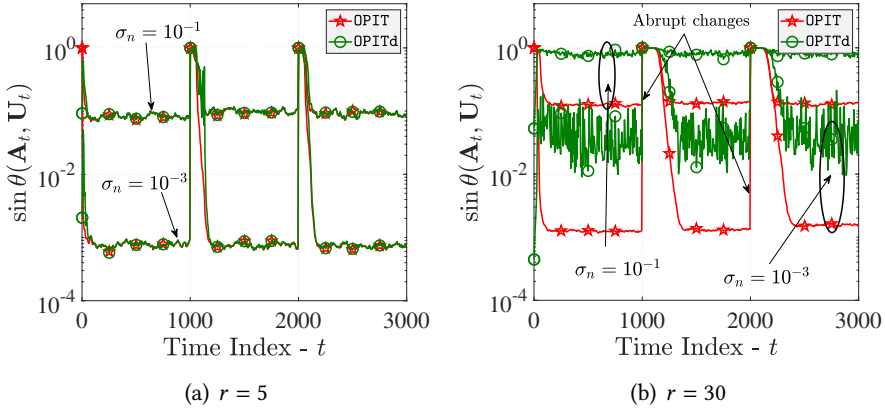
Figure 4.7: Effect of the target rank r on performance of OPITd: dimension $n = 100$, snapshots $T = 3000$, time-varying factor $\varepsilon = 10^{-3}$, sparsity level $\omega_{sparse} = 90\%$, forgetting factor $\beta = 0.97$, and two abrupt changes at $t = 1000$ and $t = 2000$.

(a) $\omega_{sparse} = 10\%$ (b) $\omega_{sparse} = 50\%$
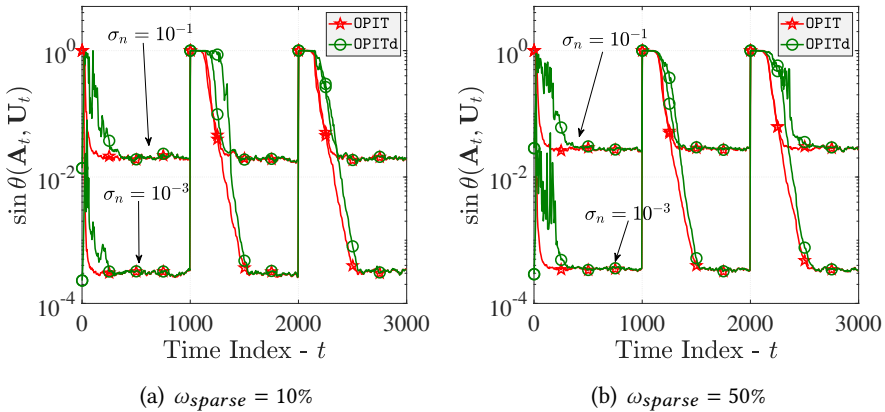
Figure 4.8: Effect of the sparsity level ω_{sparse} on performance of OPITd: dimension $n = 100$, rank $r = 20$, snapshots $T = 3000$, time-varying factor $\varepsilon = 10^{-3}$, forgetting factor $\beta = 0.97$, and two abrupt changes at $t = 1000$ and $t = 2000$.

4.5.2   Experiments with Real Video Data

In this task, four different video sequences are used to illustrate the effectiveness and efficiency of OPIT for real data, including "Lobby", "Hall", "Highway", and "Park" whose details are reported in Tab. 1, (see Fig. 4.9 for an illustration). We here compared the video tracking ability of OPIT with the state-

a) Lobby   b) Hall   c) Highway   d) Park
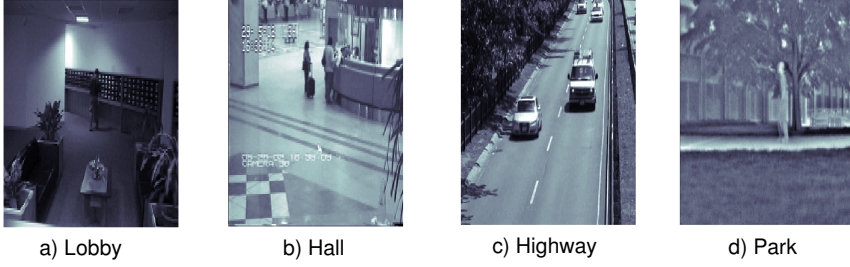
Figure 4.9: Four video sequences used in this chapter.

(a) "Lobby"

(b) "Highway"
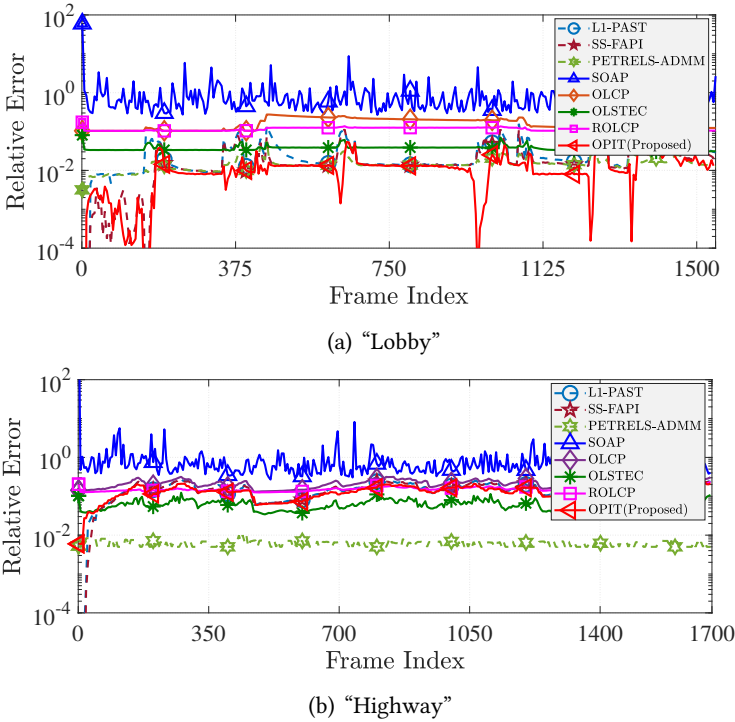
Figure 4.10: Tracking ability of algorithms on the video datasets.

of-the-art subspace tracking algorithms (i.e., ℓ_1-PAST, SS-FAPI, and PETRELS-ADMM [25]) and tensor tracking algorithms (i.e., SOAP [174], OLCP [175], OLSTEC [176], and ROLCP [33]). In order to apply these subspace tracking algorithms to the video sequences, each video frame of size $I \times J$ was reshaped into a $IJ \times 1$ vector. Following the studies on video tracking in [25] and [33], the tensor rank and subspace rank were set to 10 for all simulations.

Simulation results are shown statistically in Tab. 7.1 and graphically in

	Dataset	"Lobby"		"Hall"		"Highway"		"Park"	
Size	Tensor-based	$128 \times 160 \times 1546$		$174 \times 144 \times 3584$		$320 \times 240 \times 1700$		$288 \times 352 \times 600$	
	Matrix-based	20480×1546		25056×3584		76800×1700		101376×600	
	Evaluation metrics	time(s)	error	time(s)	error	time(s)	error	time(s)	error
Tensor	SOAP	14.29	0.842	21.72	0.989	39.89	0.821	21.34	0.789
	OLCP	10.50	0.161	19.98	0.154	27.07	0.219	14.19	0.096
	OLSTEC	44.25	0.037	92.82	0.041	130.1	0.064	53.13	**0.032**
	ROLCP	**4.32**	0.114	**10.74**	0.120	**11.45**	0.154	**4.47**	0.086
Subspace	PETRELS-ADMM	118.4	0.015	305.5	**0.018**	452.6	**0.009**	203.6	**0.032**
	ℓ_1-PAST	14.11	0.031	33.73	0.101	46.78	0.159	19.21	0.058
	SS-FAPI	12.99	0.023	32.72	0.100	46.37	0.160	17.56	0.056
	OPIT ($W = 1$)	16.32	**0.013**	50.78	0.056	56.78	0.102	26.94	0.042
	OPIT ($W = \lfloor \log(IJ) \rceil$)	**1.89**	0.021	**5.62**	0.086	**6.05**	0.141	**2.83**	0.057

Table 4.1: Runtime and averaged relative error of adaptive algorithms on tracking the four video sequences.

Fig. 4.10. As can be seen that OPIT provided a competitive estimation accuracy as compared to PETRELS-ADMM while its runtime was much faster than that of the ADMM-based tracking algorithm. Indeed, OPIT had a better performance than PETRELS-ADMM on the "Lobby" data, see Fig. 4.10(a). Also, OPIT outperformed most tracking algorithms, apart from PETRELS-ADMM. With respect to runtime, ROLCP was the fastest "one-pass" tracking algorithm, several times faster than the second-best. Interestingly, our algorithm is also designed for handling a block of multiple incoming samples at each time (i.e. the length of window $W > 1$). When $W = \lfloor \log(IJ) \rceil$, OPIT was even faster than ROLCP while still retaining a reasonable video tracking accuracy.

4.6   Conclusions

In this chapter, we have proposed a new provable OPIT algorithm which is fully capable of tracking the sparse principal subspace over time in both classical regime and high-dimension, low-sample-size regime. OPIT provides a competitive performance in terms of both subspace estimation accuracy and convergence rate in the classical regime, especially when the SNR level is high. In high dimensions, OPIT outperforms other sparse subspace tracking algorithms, its estimation accuracy is much better than that of the second-

best, SS-FAPI. Besides, a fast variant of OPIT has been obtained using deflation called OPITd. Its computational complexity and memory storage are linear to the input size and they are lower than that of OPIT. Simulations carried out on real video sequences indicated that the proposed method has potential for real applications.

4.7 Appendix

4.7.1 Appendix A: Proof of Lemma 1

Because $\mathbf{U}_{t,\mathcal{F}}$ is the Q-factor of $\mathbf{S}_t$, we obtain $\theta(\mathbf{A}, \mathbf{U}_{t,\mathcal{F}}) = \theta(\mathbf{A}, \mathbf{S}_t)$ and hence

$$\tan\theta(\mathbf{A}, \mathbf{U}_{t,\mathcal{F}}) = \max_{\|\mathbf{v}\|_2=1} \left\{ f(\mathbf{v}) = \frac{\left\|\mathbf{A}_\perp^\top \mathbf{S}_t \mathbf{v}\right\|_2}{\left\|\mathbf{A}^\top \mathbf{S}_t \mathbf{v}\right\|_2} \right\}. \tag{4.33}$$

For any vector $\mathbf{v} \in \mathbb{R}^{r\times 1}$ and $\|\mathbf{v}\|_2 = 1$, we can rewrite $f(\mathbf{v})$ in (4.33) as follows

$$
\begin{aligned}
f(\mathbf{v}) &= \frac{\left\|\mathbf{A}_\perp^\top(\mathbf{C} + \Delta\mathbf{C}_t)\mathbf{U}_{t-1}\mathbf{v}\right\|_2}{\left\|\mathbf{A}^\top(\mathbf{C} + \Delta\mathbf{C}_t)\mathbf{U}_{t-1}\mathbf{v}\right\|_2} = \frac{\left\|\mathbf{A}_\perp^\top(\sigma_x^2\mathbf{A}\mathbf{A}^\top + \sigma_n^2\mathbf{I}_N + \Delta\mathbf{C}_t)\mathbf{U}_{t-1}\mathbf{v}\right\|_2}{\left\|\mathbf{A}^\top(\sigma_x^2\mathbf{A}\mathbf{A}^\top + \sigma_n^2\mathbf{I}_N + \Delta\mathbf{C}_t)\mathbf{U}_{t-1}\mathbf{v}\right\|_2} \\
&\stackrel{(i)}{=} \frac{\left\|\sigma_n^2\mathbf{A}_\perp^\top\mathbf{U}_{t-1}\mathbf{v} + \mathbf{A}_\perp^\top\Delta\mathbf{C}_t\mathbf{U}_{t-1}\mathbf{v}\right\|_2}{\left\|(\sigma_x^2 + \sigma_n^2)\mathbf{A}^\top\mathbf{U}_{t-1}\mathbf{v} + \mathbf{A}^\top\Delta\mathbf{C}_t\mathbf{U}_{t-1}\mathbf{v}\right\|_2} \\
&\stackrel{(ii)}{\leq} \frac{\sigma_n^2\left\|\mathbf{A}_\perp^\top\mathbf{U}_{t-1}\right\|_2 + \left\|\mathbf{A}_\perp^\top\Delta\mathbf{C}_t\mathbf{U}_{t-1}\right\|_2}{(\sigma_x^2 + \sigma_n^2)\left\|\mathbf{A}^\top\mathbf{U}_{t-1}\right\|_2 - \left\|\mathbf{A}^\top\Delta\mathbf{C}_t\mathbf{U}_{t-1}\right\|_2} \\
&\stackrel{(iii)}{\leq} \frac{\sigma_n^2\left\|\mathbf{A}_\perp^\top\mathbf{U}_{t-1}\right\|_2 + \left\|\Delta\mathbf{C}_t\right\|_2}{(\sigma_x^2 + \sigma_n^2)\sqrt{1 - \left\|\mathbf{A}_\perp^\top\mathbf{U}_{t-1}\right\|_2^2} - \left\|\Delta\mathbf{C}_t\right\|_2}.
\end{aligned}
\tag{4.34}
$$

Here, (i) is due to $\mathbf{A}_\perp^\top\mathbf{A} = \mathbf{0}$ (orthogonal complement); (ii) uses the inequality $\|\mathbf{P}\|_2 - \|\mathbf{Q}\|_2 \leq \|\mathbf{P} + \mathbf{Q}\|_2 \leq \|\mathbf{P}\|_2 + \|\mathbf{Q}\|_2$, $\forall\mathbf{P}, \mathbf{Q}$ of the same size; and (iii) is derived from the following facts: $\|\mathbf{P}\Delta\mathbf{C}_t\|_2 \leq \|\mathbf{P}\|_2\|\Delta\mathbf{C}_t\|_2$, $\|\mathbf{A}\|_2 = \|\mathbf{A}_\perp\|_2 = \|\mathbf{U}_{t-1}\|_2 = 1$, and

$$\lambda_{\min}^2(\mathbf{A}^\top\mathbf{U}_{t-1}) + \lambda_{\max}^2(\mathbf{A}_\perp^\top\mathbf{U}_{t-1}) = 1, \tag{4.35}$$

where $\lambda_{\max}(\mathbf{P})$ and $\lambda_{\min}(\mathbf{P})$ represent the largest and smallest singular value of $\mathbf{P}$, respectively.

Indeed, the relation (4.35) leads to

$$
\begin{aligned}
\left\|\mathbf{A}^\top\mathbf{U}_{t-1}\right\|_2 &= \lambda_{\max}(\mathbf{A}^\top\mathbf{U}_{t-1}) \geq \lambda_{\min}(\mathbf{A}^\top\mathbf{U}_{t-1}) \\
&= \sqrt{1 - \lambda_{\max}^2(\mathbf{A}_\perp^\top\mathbf{U}_{t-1})} = \sqrt{1 - \left\|\mathbf{A}_\perp^\top\mathbf{U}_{t-1}\right\|_2^2},
\end{aligned}
\tag{4.36}
$$

and thus, (iii) follows.

In parallel, it is well known that $\sin \psi = 1/\sqrt{1 + \tan^{-2} \psi} \; \forall \psi \in [0, \pi/2]$ and $h(x) = 1/\sqrt{1 + x^{-2}}$ is an increasing function in the domain $(0, \infty)$, i.e. $x_1 \le x_2$ implies $h(x_1) \le h(x_2)$. Accordingly, we obtain

$$
\left\| A_\perp^\top U_{t,\mathcal{F}} \right\|_2 \le \frac{1}{\sqrt{1 + \left[\max_v f(v) \right]^{-2}}}
$$

$$
= \frac{\sigma_n^2 \left\| A_\perp^\top U_{t-1} \right\|_2 + \left\| \Delta C_t \right\|_2}{\left(\left[(\sigma_x^2 + \sigma_n^2) \sqrt{1 - \left\| A_\perp^\top U_{t-1} \right\|^2} - \left\| \Delta C_t \right\|_2 \right]^2 + \left[\sigma_n^2 \left\| A_\perp^\top U_{t-1} \right\|_2 + \left\| \Delta C_t \right\|_2 \right]^2 \right)^{1/2}}. \tag{4.37}
$$

It ends the proof.

4.7.2 Appendix B: Proof of Lemma 2

We first recast $\left\| U_{t,\perp}^\top U_{t,\mathcal{F}} \right\|_2$ into the following form

$$
\left\| U_{t,\perp}^\top U_{t,\mathcal{F}} \right\|_2 = \left\| U_{t,\mathcal{F},\perp}^\top U_t \right\|_2 = \left\| U_{t,\mathcal{F},\perp}^\top (U_t - U_{t,\mathcal{F}}) \right\|_2 = \left\| U_{t,\mathcal{F},\perp}^\top \Delta U_t \right\|_2. \tag{4.38}
$$

Under the following condition

$$
(1 + \sqrt{2}) \kappa(S_t) \left\| S_t - \hat{S}_t \right\|_F < \left\| S_t \right\|_2, \tag{4.39}
$$

where $\Delta S_t = S_t - \hat{S}_t$ and $\kappa(S_t) = \| S_t^\# \|_2 \| S_t \|_2$, we can bound this distance as follows

$$
\left\| U_{t,\mathcal{F},\perp}^\top \Delta U_t \right\|_2 \le \left\| U_{t,\mathcal{F},\perp}^\top \Delta U_t \right\|_F \overset{(i)}{\le} \frac{\kappa(S_t) \dfrac{\| U_{t,\mathcal{F},\perp}^\top \Delta S_t \|_F}{\| S_t \|_2}}{1 - (1 + \sqrt{2}) \kappa(S_t) \dfrac{\| \Delta S_t \|_F}{\| S_t \|_2}}
$$

$$
\overset{(ii)}{\le} \frac{\| \Delta S_t \|_F}{\lambda_{\min}(S_t) - (1 + \sqrt{2}) \| \Delta S_t \|_F}. \tag{4.40}
$$

Here, (i) follows immediately the perturbation theory for QR decomposition [177, Theorem 3.1] and (ii) is obtained from the facts that $\| U_{t,\mathcal{F},\perp} \|_2 = 1$, $\| PQ \|_F \le \| P \|_2 \| Q \|_F$, and $\| P^\# \|_2 = \lambda_{\min}^{-1}(P) \; \forall P, Q$ of suitable sizes.

We also know that there always exists two coefficient matrices $H_t \in \mathbb{R}^{r \times r}$ and $K_t \in \mathbb{R}^{(n-r) \times r}$ satisfying $U_{t-1} = A H_t + A_\perp K_t$ (i.e. projection of U_{t-1} onto the subspace A) and

$$
\lambda_{\max}(H_t) = \| A^\top U_{t-1} \|_2, \; \lambda_{\min}(H_t) = \sqrt{1 - \| A_\perp^\top U_{t-1} \|_2^2}, \tag{4.41}
$$

$$
\lambda_{\max}(K_t) = \| A_\perp^\top U_{t-1} \|_2, \; \lambda_{\min}(K_t) = \sqrt{1 - \| A^\top U_{t-1} \|_2^2}.
$$

Accordingly, we can express $\mathbf{S}_t$ by

$$\mathbf{S}_t = \mathbf{C}\mathbf{U}_{t-1} + \Delta\mathbf{C}_t\mathbf{U}_{t-1} = (\mathbf{A}\Sigma_x\mathbf{A}^\top + \sigma_n^2\mathbf{I}_n)(\mathbf{A}\mathbf{H}_t + \mathbf{A}_\perp\mathbf{K}_t) + \Delta\mathbf{C}_t\mathbf{U}_{t-1}$$
$$= \mathbf{A}(\sigma_x^2\mathbf{I}_r + \sigma_n^2\mathbf{I}_r)\mathbf{H}_t + \sigma_n^2\mathbf{A}_\perp\mathbf{K}_t + \Delta\mathbf{C}_t\mathbf{U}_{t-1}. \quad (4.42)$$

Thanks to the fact that $\lambda_i(\mathbf{P} + \mathbf{Q}) \geq \lambda_i(\mathbf{P}) - \lambda_{\max}(\mathbf{Q}) \ \forall \mathbf{P}, \mathbf{Q}$ of the same size, the lower bound on $\lambda_{\min}(\mathbf{S}_t)$ is given by

$$\lambda_{\min}(\mathbf{S}_t) \geq \lambda_{\min}\big((\sigma_x^2 + \sigma_n^2)\mathbf{A}\mathbf{H}_t\big) - \lambda_{\max}\big(\sigma_n^2\mathbf{A}_\perp\mathbf{K}_t\big) - \lambda_{\max}\big(\Delta\mathbf{C}_t\mathbf{U}_{t-1}\big)$$
$$\geq (\sigma_x^2 + \sigma_n^2)\lambda_{\min}(\mathbf{H}_t) - \sigma_n^2\lambda_{\max}(\mathbf{K}_t) - \|\Delta\mathbf{C}_t\|_2$$
$$= (\sigma_x^2 + \sigma_n^2)\sqrt{1 - \|\mathbf{A}_\perp^\top\mathbf{U}_{t-1}\|_2^2} - \sigma_n^2\|\mathbf{A}_\perp^\top\mathbf{U}_{t-1}\|_2 - \|\Delta\mathbf{C}_t\|_2, \quad (4.43)$$

In what follows, we derive an upper bound on $\|\Delta\mathbf{S}_t\|_F$. For short, let us denote the support of $\mathbf{A}$, $\mathbf{U}_{t-1}$, and $\mathbf{U}_t$ by $\mathcal{T}_A$, $\mathcal{T}_{t-1}$, and $\mathcal{T}_t$, respectively, and $\mathcal{S}_t = \mathcal{T}_A \cup \mathcal{T}_{t-1} \cup \mathcal{T}_t$. Here, it is easy to verify that $\mathbf{S}_{t,\mathcal{S}_t} = \mathbf{C}_{t,\mathcal{S}_t \times \mathcal{S}_t}\mathbf{U}_{t-1}$ and $\hat{\mathbf{S}}_t = \mathbf{S}_{t,\mathcal{T}_t} = \tau(\mathbf{S}_{t,\mathcal{S}_t}, k)$. Accordingly, we can bound $\|\Delta\mathbf{S}_t\|_F$ as follows

$$\big\|\Delta\mathbf{S}_t\big\|_F = \big\|\mathbf{S}_{t,\mathcal{S}_t} - \mathbf{S}_{t,\mathcal{T}_t}\big\|_F \overset{(i)}{\leq} \big\|\mathbf{S}_{t,\mathcal{S}_t} - \mathbf{S}_{t,\mathcal{T}_A}\big\|_F$$
$$= \big\|\sigma_n^2\mathbf{A}_\perp\mathbf{K}_t + \Delta\mathbf{C}_t\mathbf{U}_{t-1}\big\|_F$$
$$\leq \sqrt{r}\big\|\sigma_n^2\mathbf{A}_\perp\mathbf{K}_t + \Delta\mathbf{C}_t\mathbf{U}_{t-1}\big\|_2 \leq \sqrt{r}\big(\sigma_n^2\|\mathbf{K}_t\|_2 + \|\Delta\mathbf{C}_t\|_2\big)$$
$$= \sqrt{r}\big(\sigma_n^2\|\mathbf{A}_\perp^\top\mathbf{U}_{t-1}\|_2 + \|\Delta\mathbf{C}_t\|_2\big), \quad (4.44)$$

where (i) is due to $|\mathcal{T}_t| \geq |\mathcal{T}_A| \ \forall t$ (i.e. $|\mathcal{S}_t \setminus \mathcal{T}_t| \leq |\mathcal{S}_t \setminus \mathcal{T}_A|$), thanks the thresholding operator $\tau(.)$ with $k/n \geq \omega_{sparse}$.

In parallel, we can rewrite the sufficient and necessary condition (4.39) as

$$(1 + \sqrt{2})\big\|\mathbf{S}_t^\#\big\|_2\big\|\Delta\mathbf{S}_t\big\|_F \leq 1. \quad (4.45)$$

Since $\|\mathbf{S}_t^\#\|_2 = \lambda_{\min}^{-1}(\mathbf{S}_t)$, substituting (4.43) for $\|\mathbf{S}_t^\#\|_2$ and (4.44) for $\|\Delta\mathbf{S}_t\|_F$ results in

$$\frac{\sigma_n^2\big\|\mathbf{A}_\perp^\top\mathbf{U}_{t-1}\big\|_2 + \|\Delta\mathbf{C}_t\|_2}{(\sigma_x^2 + \sigma_n^2)\sqrt{1 - \big\|\mathbf{A}_\perp^\top\mathbf{U}_{t-1}\big\|^2}} \leq \frac{\sqrt{2} - 1}{\sqrt{r} - 1 + \sqrt{2}}. \quad (4.46)$$

Under the condition (4.46), the upper bound on $\|\mathbf{U}_{t,\perp}^{\top}\mathbf{U}_{t,\mathcal{F}}\|_2$ is

$$
\left\|\mathbf{U}_{t,\perp}^{\top}\mathbf{U}_{t,\mathcal{F}}\right\|_2 \leq \frac{\sqrt{r}\left(\sigma_n^2\left\|\mathbf{A}_{\perp}^{\top}\mathbf{U}_{t-1}\right\|_2 + \|\Delta\mathbf{C}_t\|_2\right)}{\left(\left(\sigma_x^2 + \sigma_n^2\right)\sqrt{1 - \left\|\mathbf{A}_{\perp}^{\top}\mathbf{U}_{t-1}\right\|_2^2} - \sigma_n^2\left\|\mathbf{A}_{\perp}^{\top}\mathbf{U}_{t-1}\right\|_2 - \right.}
$$

$$
{\left. - \|\Delta\mathbf{C}_t\|_2 - \sqrt{r}\left(1 + \sqrt{2}\right)\left(\sigma_n^2\left\|\mathbf{A}_{\perp}^{\top}\mathbf{U}_{t-1}\right\|_2 + \|\Delta\mathbf{C}_t\|_2\right)\right)}
$$

$$
= \frac{\sqrt{r}\left(\sigma_n^2\left\|\mathbf{A}_{\perp}^{\top}\mathbf{U}_{t-1}\right\|_2 + \|\Delta\mathbf{C}_t\|_2\right)}{\left(\left(\sigma_x^2 + \sigma_n^2\right)\sqrt{1 - \left\|\mathbf{A}_{\perp}^{\top}\mathbf{U}_{t-1}\right\|_2^2} - \left(1 + \sqrt{r}(1 + \sqrt{2})\right)\times \right.}, \quad (4.47)
$$

$$
{\left. \times \left(\sigma_n^2\left\|\mathbf{A}_{\perp}^{\top}\mathbf{U}_{t-1}\right\|_2 + \|\Delta\mathbf{C}_t\|_2\right)\right)}
$$

thanks to (4.40). It ends the proof.

4.7.3   Appendix C: Proof of Lemma 3

We begin the proof with the following proposition:

> **Proposition 10** *Given two sets of random variable vectors $\{\mathbf{a}_i\}_{i=1}^N$ and $\{\mathbf{b}_i\}_{i=1}^N$ where $\mathbf{a}_i \overset{i.i.d.}{\sim} \mathcal{N}(\mathbf{0}, \sigma_a^2\mathbf{I}_n)$, $\mathbf{b}_i \overset{i.i.d.}{\sim} \mathcal{N}(\mathbf{0}, \sigma_b^2\mathbf{I}_m)$, and $\mathbf{a}_i$ is independent of $\mathbf{b}_j, \forall i, j$. The following inequality holds with a probability at least $1 - \delta$:*
>
> $$
> \left\|\frac{1}{N}\sum_{i=1}^{N}\mathbf{a}_i\mathbf{b}_i^{\top}\right\|_2 \leq C\sigma_a\sigma_b\sqrt{\log(2/\delta)\frac{\max\{n, m\}}{N}}. \quad (4.48)
> $$
>
> *where $0 < \delta \ll 1$ and $C > 0$ is a universal positive number.*

Proof. Its proof follows immediately Lemma 15 in [178].
Since $\mathbf{x}_i = \mathbf{A}\mathbf{w}_i + \mathbf{n}_i$, we always have

$$
\|\Delta\mathbf{C}_t\|_2 = \left\|\frac{1}{tW}\sum_{i=1}^{tW}\mathbf{x}_i\mathbf{x}_i^{\top} - \mathbf{C}\right\|_2
$$

$$
= \left\|\frac{1}{tW}\sum_{i=1}^{tW}\left(\mathbf{A}\mathbf{w}_i\mathbf{w}_i^{\top}\mathbf{A}^{\top} + \mathbf{n}_i\mathbf{n}_i^{\top} + \mathbf{A}\mathbf{w}_i\mathbf{n}_i^{\top} + \mathbf{n}_i\mathbf{w}_i^{\top}\mathbf{A}^{\top}\right) - \sigma_x^2\mathbf{A}\mathbf{A}^{\top} - \sigma_n^2\mathbf{I}_n\right\|_2
$$

$$\leq \left\| \mathbf{A}\left(\frac{1}{tW}\sum_{i=1}^{tW}\mathbf{w}_i\mathbf{w}_i^\top - \sigma_x^2\mathbf{I}_r\right)\mathbf{A}^\top \right\|_2 + \left\| \frac{1}{tW}\sum_{i=1}^{tW}\mathbf{n}_i\mathbf{n}_i^\top - \sigma_n^2\mathbf{I}_N \right\|_2$$

$$+ 2\left\| \mathbf{A}\left(\frac{1}{tW}\sum_{i=1}^{tW}\mathbf{w}_i\mathbf{n}_i^\top\right) \right\|_2$$

$$\leq \|\mathbf{A}\|_2^2 \left\| \frac{1}{tW}\sum_{i=1}^{tW}\mathbf{w}_i\mathbf{w}_i^\top - \sigma_x^2\mathbf{I}_r \right\|_2 + \left\| \frac{1}{tW}\sum_{i=1}^{tW}\mathbf{n}_i\mathbf{n}_i^\top - \sigma_n^2\mathbf{I}_n \right\|_2$$

$$+ 2\|\mathbf{A}\|_2 \left\| \frac{1}{tW}\sum_{i=1}^{tW}\mathbf{w}_i\mathbf{n}_i^\top \right\|_2, \quad (4.49)$$

thanks to the inequality $\|\mathbf{PQ}\|_2 \leq \|\mathbf{P}\|_2\|\mathbf{Q}\|_2$ for all $\mathbf{P}$ and $\mathbf{Q}$ of suitable sizes. Accordingly, with a probability at least $1 - \delta$ ($0 < \delta \ll 1$), three components in the right hand side of (4.49) are respectively bounded by

$$\left\| \frac{1}{tW}\sum_{i=1}^{tW}\mathbf{w}_i\mathbf{w}_i^\top - \sigma_x^2\mathbf{I}_r \right\|_2 \leq C_1\sqrt{\log(2/\delta)}\sigma_x^2\sqrt{\frac{r}{tW}}, \quad (4.50)$$

$$\left\| \frac{1}{tW}\sum_{i=1}^{tW}\mathbf{n}_i\mathbf{n}_i^\top - \sigma_n^2\mathbf{I}_n \right\|_2 \leq C_2\sqrt{\log(2/\delta)}\sigma_n^2\sqrt{\frac{n}{tW}}, \quad (4.51)$$

$$\left\| \frac{1}{tW}\sum_{i=1}^{tW}\mathbf{w}_i\mathbf{n}_i^\top \right\|_2 \leq C_3\sqrt{\log(2/\delta)}\sigma_x\sigma_n\sqrt{\frac{n}{tW}}, \quad (4.52)$$

where C_1, C_2, C_3 are universal positive parameters, thanks to Proposition 10 and [24, Proposition 2.1]. As a result, we obtain

$$\left\| \Delta\mathbf{C}_t \right\|_2 \leq c_\delta\left(\sigma_x^2\sqrt{\frac{r}{tW}} + \left(2\sigma_n\sigma_x + \sigma_n^2\right)\sqrt{\frac{n}{tW}} \right), \quad (4.53)$$

where $c_\delta = \max\left\{C_1, C_2, C_3\right\}\sqrt{\log(2/\delta)}$. It ends the proof.

4.7.4 Appendix D: Proof of Lemma 4

We first use proof by induction to prove $d_t \leq \omega_0 = \max\{d_0, \epsilon\}$. Particularly, we already have the base case of $d_0 \leq \omega_0$. In the induction step, we suppose $d_{t-1} \leq \omega_0$ and then prove $d_t \leq \omega_0$ still holds. After that, we indicate that $d_t \leq \epsilon$ is achievable when the two conditions (4.18) and (4.19) are met.

Thanks to Lemma 3, when t satisfies (4.18), i.e.,

$$t \geq \frac{C\log(2/\delta)r^2}{W\epsilon^2\rho^2}\left(\sqrt{r} + \left(\frac{\sigma_n^2}{\sigma_x^2} + 2\frac{\sigma_n}{\sigma_x}\right)\sqrt{n} \right)^2, \quad (4.54)$$

we obtain $\|\Delta \mathbf{C}_t\|_2 \le r^{-1}\rho\sigma_x^2\epsilon$ with $0 < \rho \le r$. In what follows, two case studies $d_{t-1} \ge \epsilon$ and $d_{t-1} \le \epsilon$ are investigated.

Case 1: When $d_{t-1} \ge \epsilon$, i.e., $\|\Delta \mathbf{C}_t\|_2 \le r^{-1}\rho\sigma_x^2 d_{t-1}$.

We can rewrite $\|\mathbf{A}_\perp^\top \mathbf{U}_{t,\mathcal{F}}\|_2$ as follows

$$\left\|\mathbf{A}_\perp^\top \mathbf{U}_{t,\mathcal{F}}\right\|_2 \le \frac{(\sigma_n^2 + r^{-1}\rho\sigma_x^2)d_{t-1}}{\left(\left[(\sigma_n^2 + \sigma_x^2)\sqrt{1-d_{t-1}^2} - r^{-1}\rho\sigma_x^2 d_{t-1}\right]^2 + \atop + (\sigma_n^2 + \sigma_x^2\rho/r)^2 d_{t-1}^2\right)^{1/2}}$$

$$\overset{(i)}{\le} \frac{(\sigma_n^2 + r^{-1}\rho\sigma_x^2)d_{t-1}}{\left(\left[(\sigma_n^2 + \sigma_x^2)\sqrt{1-\omega_0^2} - r^{-1}\rho\sigma_x^2\omega_0\right]^2 + \atop + (\sigma_n^2 + r^{-1}\rho\sigma_x^2)^2\omega_0^2\right)^{1/2}}$$

$$\overset{(ii)}{\le} \frac{(\sigma_n^2 + r^{-1}\rho\sigma_x^2)d_{t-1}}{\left((1+\gamma^2 r^2)\sigma_n^4 + (1-\rho\gamma)^2\sigma_x^4 + \atop + 2(1-\rho\gamma+\gamma^2 r^2)\sigma_x^2\sigma_n^2\right)^{1/2}\sqrt{1-\omega_0^2}}. \qquad (4.55)$$

Here, (i) is obtained from the fact that $g(x) = \left((a\sqrt{1-x^2} - bx)^2 + cx^2\right)^{-1/2}$ is an increasing function in the range $[0, \sqrt{2}/2]$ where a, b, and c are defined therein[8] and (ii) is simple due to the fact that there always exists a small parameter $\gamma > 0$ such that $\rho\gamma < 1$ and $\omega_0 \le \gamma r\sqrt{1-\omega_0^2}$.

In the similar way, we obtain the following upper bound on $\|\mathbf{U}_{t,\perp}^\top \mathbf{U}_{t,\mathcal{F}}\|_2$:

$$\left\|\mathbf{U}_{t,\perp}^\top \mathbf{U}_{t,\mathcal{F}}\right\|_2 \le \frac{\sqrt{r}(\sigma_n^2 + r^{-1}\rho\sigma_x^2)d_{t-1}}{(\sigma_x^2 + \sigma_n^2)\sqrt{1-d_t^2} - \left(1 + \sqrt{r}(1+\sqrt{2})\right)\times \atop \times (\sigma_n^2 + r^{-1}\rho\sigma_x^2)d_{t-1}}$$

$$\overset{(i)}{\le} \frac{\sqrt{r}(\sigma_n^2 + r^{-1}\rho\sigma_x^2)d_{t-1}}{(\sigma_x^2 + \sigma_n^2)\sqrt{1-\omega_0^2} - \left(1 + \sqrt{r}(1+\sqrt{2})\right)(\sigma_n^2 + r^{-1}\rho\sigma_x^2)\omega_0}$$

$$\overset{(ii)}{\le} \frac{\sqrt{r}(\sigma_n^2 + r^{-1}\rho\sigma_x^2)}{(\sigma_x^2 + \sigma_n^2)(1-\varrho)\sqrt{1-\omega_0^2}}d_{t-1}, \qquad (4.56)$$

---

[8]Writing $x = \sin y$, the domain of y is $[0, \pi/4]$. Here, we can recast $g(x)$ into $g(y) = \left((a\cos y - b\sin y)^2 + c\sin^2 y\right)^{-1/2}$. The derivative $g'(y)$ is given by

$$g'(y) = 0.5\left((a\cos y - b\sin y)^2 + c\sin^2 y\right)^{-3/2}\left((a^2 - b^2 - c)\sin(2y) + ab\cos(2y)\right).$$

Since $a^2 - b^2 > c$ by their definition, $g'(y) > 0 \; \forall y \in [0, \pi/4]$ and hence $g'(x) = g'(y)dy/dx = g'(y)/\sqrt{1-x^2} > 0 \; \forall x \in [0, \sqrt{2}/2]$. Accordingly, $d_{t-1} \le \omega_0 \le \sqrt{2}/2$ implies $g(d_{t-1}) \le g(\omega_0)$ which (i) then follows.

where $\varrho = \gamma\big(1 + \sqrt{r}(1 + \sqrt{2})(r\sigma_n^2 + \rho\sigma_x^2)\big)(\sigma_x^2 + \sigma_n^2)^{-1}$. Specifically, (i) is due to the increasing property of $z(x) = (a\sqrt{1 - x^2} - bx)^{-1}$, and (ii) thanks to $\omega_0 \le \gamma r\sqrt{1 - \omega_0^2}$.

Thanks to (4.55) and (4.56), we obtain

$$d_t \le \big\|A_\perp^\top U_{t,\mathcal{F}}\big\|_2 + \big\|U_{t,\perp}^\top U_{t,\mathcal{F}}\big\|_2 \le \frac{r\sigma_n^2 + \rho\sigma_x^2}{r\xi\sqrt{1 - \omega_0^2}}d_{t-1}, \tag{4.57}$$

where

$$\xi = 0.5\max\left\{\Big((1 + \gamma^2 r^2)\sigma_n^4 + (1 - \rho\gamma)^2\sigma_x^4 + 2(1 - \rho\gamma + \gamma^2 r^2)\sigma_x^2\sigma_n^2\Big)^{1/2},\right.$$
$$\left.(\sigma_x^2 + \sigma_n^2)(1 - \varrho)/\sqrt{r}\right\}. \tag{4.58}$$

Note that in order to utilize the two bounds (6.189) and (4.56), the condition (4.46) must be satisfied which is equivalent to

$$\frac{(\sigma_n^2 + r^{-1}\rho\sigma_x^2)\omega_0}{(\sigma_x^2 + \sigma_n^2)\sqrt{1 - \omega_0^2}} \le \frac{\sqrt{2} - 1}{\sqrt{r} - 1 + \sqrt{2}}. \tag{4.59}$$

Accordingly, we obtain $\omega_0 \le \left(\frac{\alpha(r,\rho)}{1 - \alpha(r,\rho)}\right)^{1/2}$ where

$$\alpha(r, \rho) = \frac{(3 - 2\sqrt{2})(\sigma_x^2 + \sigma_n^2)^2}{\big(r + 2\sqrt{r}(\sqrt{2} - 1) + 3 - 2\sqrt{2}\big)\big(\sigma_n^2 + r^{-1}\rho\sigma_x^2\big)^2}. \tag{4.60}$$

In parallel, $\alpha(r, \rho) \ge \frac{3 - 2\sqrt{2}}{r + 2\sqrt{r}(\sqrt{2}-1)+3-2\sqrt{2}}$ for every $0 < \rho \le r$. Thus, we obtain

$$\omega_0 \le \left(\frac{3 - 2\sqrt{2}}{r + 2\sqrt{r}(\sqrt{2} - 1)}\right)^{1/2}, \tag{4.61}$$

which is exactly the condition (4.19) in Theorem 1. Moreover, there are various options of $p \in (0, r]$ satisfying $\rho\sigma_x^2 < r\xi\sqrt{1 - \omega_0^2} - r\sigma_n^2$, e.g., when the value of ρ is very close to zero. In such cases, d_t will decrease in each time t, i.e., $d_t \le d_{t-1} \le \omega_0$.

Case 2: When $d_{t-1} \le \epsilon$, applying the same arguments in Case 1, we also obtain $d_t \le \frac{r\sigma_n^2 + \rho\sigma_x^2}{r\xi\sqrt{1 - \omega_0^2}}\epsilon \le \epsilon \le \omega_0$.

To sum up, if the two conditions (4.18) and (4.19) are satisfied, then $d_t \le \max\{d_{t-1}, \epsilon\} = \omega_0$. As a result, the statement $d_t \le \epsilon$ holds if and only if

$$\left(\frac{r\sigma_n^2 + \rho\sigma_x^2}{r\xi\sqrt{1 - \omega_0^2}}\right)^{tW}\omega_0 \le \epsilon. \tag{4.62}$$

(a) $\sigma_n = \epsilon = 10^{-3}, \omega_{sparse} = 90\%$ (b) $\sigma_n = \epsilon = 10^{-2}, \omega_{sparse} = 90\%$
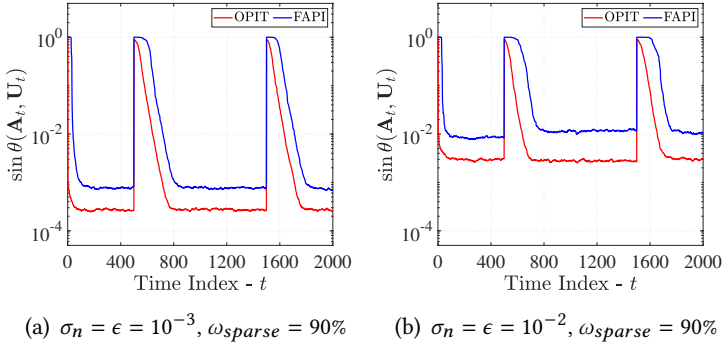
Figure 4.11: OPIT vs the best optimal power-based subspace tracker FAPI: Data dimension $n = 100$, true rank 10, number of snapshots $T = 2000$, forgetting factor $\beta = 0.97$, abrupt changes at $t = 500$ and $t = 1500$.

Specifically, (4.62) is equivalent to

$$t \geq \frac{\log(\epsilon/\omega_0)}{W\left(\log(r\sigma_n^2 + \rho\sigma_x^2) - \log(r\xi\sqrt{1 - \omega_0^2})\right)}. \tag{4.63}$$

which is lower than the bound (4.18). Therefore, we can conclude that $d_t \leq \epsilon$ holds and it ends the proof.

Appendix E: Additional Experimental Results

OPIT vs the best optimal power-based subspace tracker FAPI

Here, we illustrate that OPIT is more effective than the existing power-based subspace trackers. As it is well-documented that FAPI is the best optimal power-based subspace tracker w.r.t. both convergence rate and estimation accuracy [154], we adopt FAPI in this work. We set the data dimension $n = 100$, the true rank $r = 10$, the number of data samples $T = 2000$. Two levels of noise and time-varying factors are considered, namely $\sigma_n = \epsilon = 10^{-3}$ and $\sigma_n = \epsilon = 10^{-2}$. To assess how fast subspace trackers converge, we create two abrupt changes at $t = 500$ and $t = 1500$. To have a fair comparison, the forgetting factor β is fixed at the same value 0.97 for both OPIT and FAPI in all testing cases. Results are shown as in Fig. 4.11. We can see that OPIT yields higher subspace estimation accuracy than FAPI. When abrupt changes happen, OPIT also converges faster than FAPI.

(a) $r = 2$, $\omega_{sparse} = 50\%$

(b) $r = 2$, $\omega_{sparse} = 90\%$

(c) $r = 10$, $\omega_{sparse} = 50\%$

(d) $r = 10$, $\omega_{sparse} = 90\%$
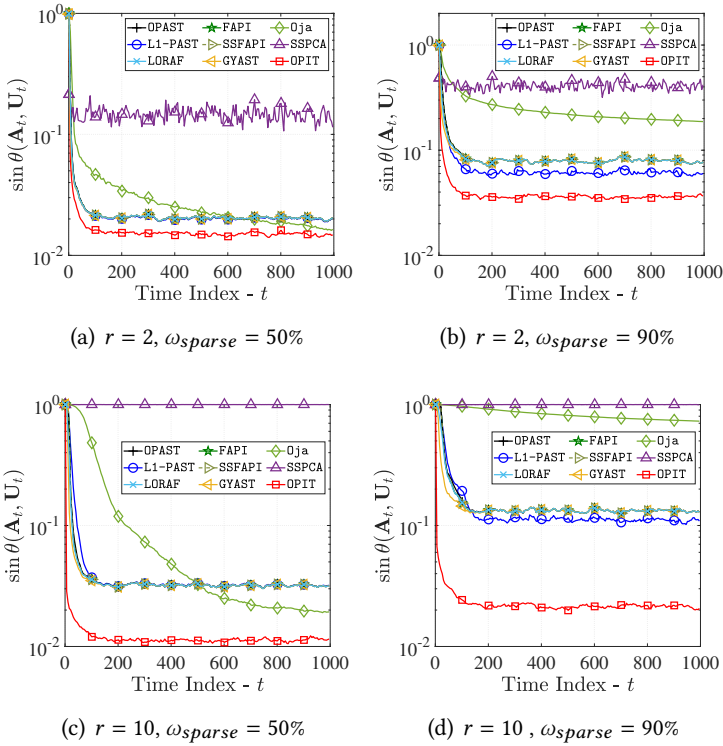
Figure 4.12: Performance comparisons between OPIT and other ST algorithms in the classical setting: dimension $n = 50$, snapshots $T = 1000$, time-varying factor $\varepsilon = 10^{-3}$, and the noise level $\sigma_n = 10^{-1}$.

OPIT vs State-of-the-art Subspace Trackers

In this subsection, we provide further performance comparison of OPIT against the state-of-the-art subspace trackers addressed in Section V.4 in the main text. Fig. 4.12 and Fig. 4.13 illustrate the experimental results in the classical regime and high dimensions when the noise level is high, i.e., $\sigma_n = 10^{-1}$. As can be seen that OPIT outperform others completely in both regimes.

OPIT vs Data Dimension and Sample Size

This subsection provides additional experimental results of OPIT to demonstrate the effectiveness of OPIT in many settings of data dimension and sample size. Please see Figs. 4-8 for details.

(a) $n = 1000$, $\omega_{sparse} = 50\%$

(b) $n = 1000$, $\omega_{sparse} = 90\%$

(c) $n = 10000$, $\omega_{sparse} = 50\%$

(d) $n = 10000$, $\omega_{sparse} = 90\%$
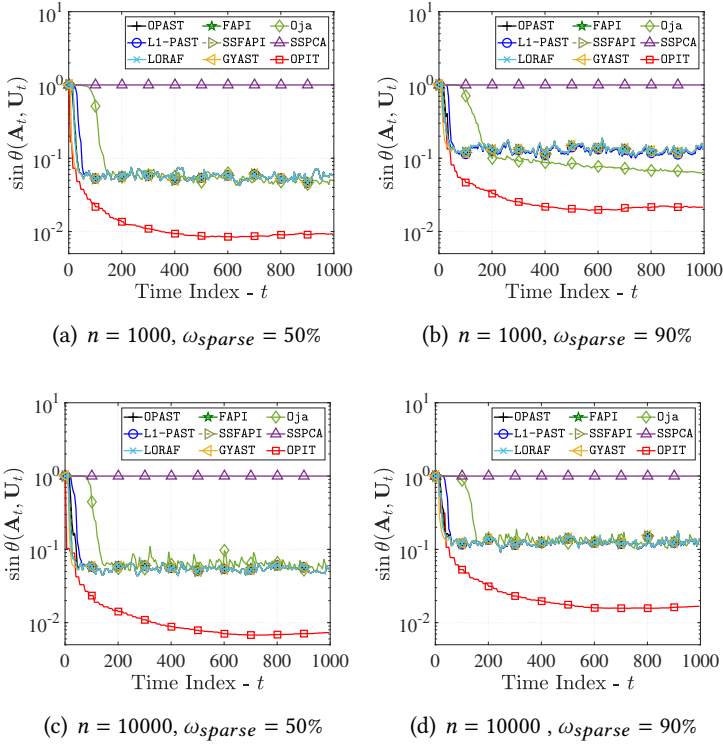
Figure 4.13: **Performance comparisons between OPIT and other SST algorithms in high dimensions: target rank $r = 10$, snapshots $T = 1000$, time-varying factor $\varepsilon = 10^{-3}$, and the noise level $\sigma_n = 10^{-1}$.**

(a) $\sigma_n = 10^{-1}$

(b) $\sigma_n = 10^{-3}$
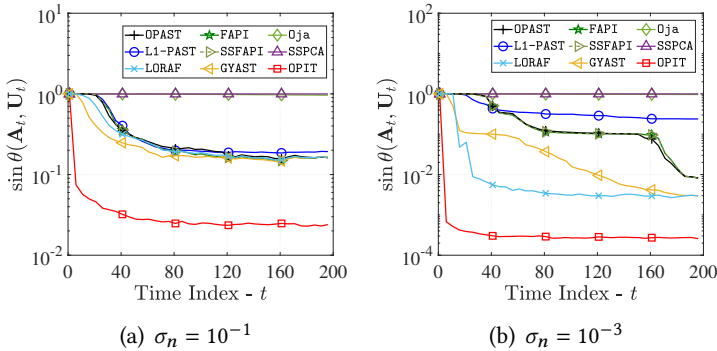
Figure 4.14: $n = 50, T = 200$: **rank $r = 10$, time-varying $\varepsilon = 10^{-3}$, sparsity 90%.**

(a) $\sigma_n = 10^{-1}$

(b) $\sigma_n = 10^{-3}$
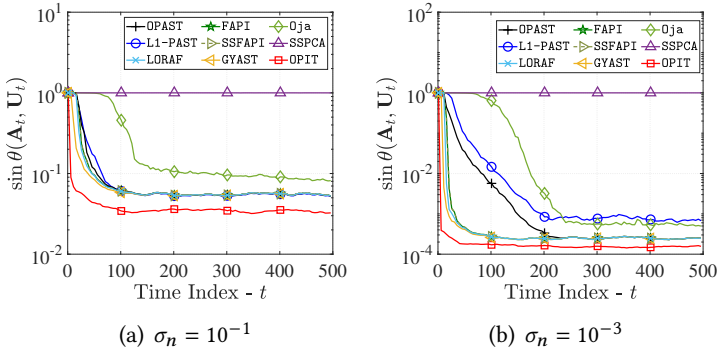
Figure 4.15: $n = 1000, T = 500$: **rank** $r = 10$, **time-varying** $\epsilon = 10^{-3}$, **sparsity 90%**

(a) $\sigma_n = 10^{-1}$

(b) $\sigma_n = 10^{-3}$
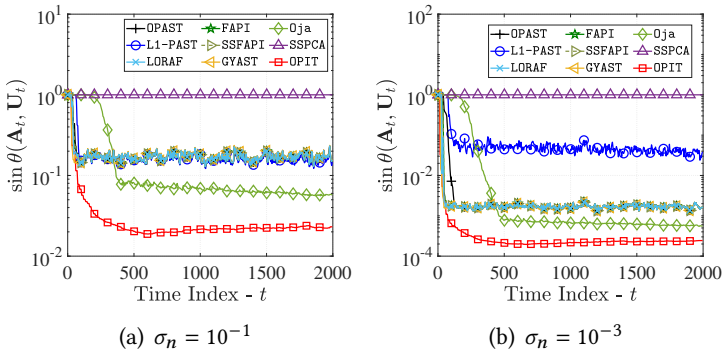
Figure 4.16: $n = 2000, T = 2000$: **rank** $r = 20$, **time-varying** $\epsilon = 10^{-3}$, **sparsity 90%**

(a) $\sigma_n = 10^{-1}$

(b) $\sigma_n = 10^{-3}$
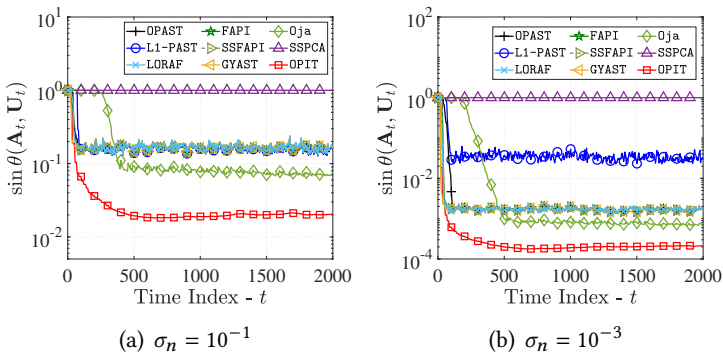
Figure 4.17: $n = 5000, T = 2000$: **rank** $r = 20$, **time-varying** $\epsilon = 10^{-3}$, **sparsity 90%**

An Overview of Tensor Tracking

<div style="text-align:right; font-size:2em">5</div>

Tensor decomposition has been demonstrated to be successful in a wide range of applications, from neuroscience and wireless communications to social networks. In an online setting, factorizing tensors derived from multidimensional data streams is however non-trivial due to several inherent problems of real-time stream processing. In recent years, many research efforts have been dedicated to developing online techniques for decomposing such tensors, resulting in significant advances in streaming tensor decomposition or tensor tracking. This topic is emerging and enriches the literature on tensor decomposition, particularly from the data stream analytics perspective. Thus, it is imperative to carry out an overview of tensor tracking to help researchers and practitioners understand its development and achievements, summarise the current trends and advances, and identify challenging problems. In this article, we provide a contemporary and comprehensive survey on different types of tensor tracking techniques. We particularly categorize the state-of-the-art methods into three main groups: streaming CP decompositions, streaming Tucker decompositions, and streaming decompositions under other tensor formats (i.e., tensor-train, t-SVD, and BTD). In each group, we further divide the existing algorithms into sub-categories based on their main optimization framework and model architectures. Finally, we present several research challenges, open problems, and potential directions of tensor tracking in the future.

5.1 Introduction

Tensor decomposition (TD) has attracted much attention from the signal processing and machine learning community [11]. As a tensor is a multiway array, it provides a natural representation for multidimensional data. Accordingly, TD which factorizes a tensor into a set of basis components (e.g., vectors, matrices, or simpler tensors) has become a popular tool for multivariate and high-dimensional data analysis. In particular, we have witnessed significant advances in TD and a rapid growth in its applications over the last two decades [13]. Several types of TD, such as CANDECOMP/PARAFAC (CP) [14], high-order SVD (HOSVD)/Tucker [15], tensor train/network [16], t-SVD [17], and block-term decomposition (BTD) [18], have been developed and successfully applied to various domains, from neuroscience [179, 180] wireless communications [181, 182] to social networks [183, 184].

The demand for (near) real-time stream processing has been increasing over the years since many modern applications (e.g., Internet-of-Things) generate massive amounts of streaming data over time and analytical insights from such data can bring several benefits, e.g., for real-time decision making [2]. As its name implies, (near) real-time stream processing needs to immediately deliver and analyse data streams upon their arrival. Since stream-

ing data grow bigger, faster, and become more complex by the time, there exist several inherent problems which are still challenging issues, such as (i) the unbounded size of streaming data, (ii) time-varying model, concept drift, or dataset shift, and (iii) uncertainty and imperfection, etc. We refer the readers to [2,3] for good surveys on data stream analysis.

When using tensors to represent data streams, TD is generally referred to as tensor tracking or adaptive/online/ streaming tensor decomposition. Specifically, factorizing a streaming tensor is nontrivial due to several computational challenges. First, as tensor streams are continuously generated, their volume grows significantly over time and possibly to infinity. Applying the conventional batch TD methods to such tensors is not possible as they require data to be stored and processed offline. Second, properties of streaming tensors (e.g., the low-rank approximation model) can vary with time in unforeseen ways. Moreover, tensor streams often happen in real-time, so retransmission of a stream is difficult, even impossible. Accordingly, batch tensor estimation and decomposition become less accurate when time passes. Last but not least, some modern applications require high-speed data acquisition systems to rapidly acquire and process massive data streams. In such a case, very fast and (near) real-time processing is highly important. However, batch TDs are often of high complexity, and hence turn out to be inefficient. These characteristics make tensor tracking much different from batch tensor decomposition and lead to several distinguishing requirements for tensor trackers, such as low latency and memory storage, high scalability, adaptation to time variation, and robustness, to name a few.

As the literature of tensor tracking has significantly expanded in recent years, it is imperative to it is imperative to conduct an extensive overview of the state-of-the-art tensor tracking algorithms to help researchers and practitioners to identify: (i) which topics in tensor tracking are significant and emerging, (ii) what kind of tracking models and related analysis techniques have already been deployed to date and how to apply them in specific tasks, and (iii) main research challenges, open problems, and potential directions of tensor tracking in the future.

5.1.1 State-of-the-art Surveys

The very first and gentle introduction to tensor and tensor decomposition was provided by Rasmus in [185] two decades ago. This reference offered a tutorial on CP/PAFRAFAC decomposition covering features, properties, methods, and applications in chemometrics. Since then, there have been many published survey papers which provided different points of view on tensor computation in the literature. We can broadly divide them into three classes, including (i) surveys on models, methods, and tools for factorizing tensors, (ii) surveys on

Table 5.1: The State-of-the-art Surveys on Tensor Decompositions and Applications

Class	Review (Year)	Objects & Topics	Key Contribution
Surveys on tensor factorization models, methods, and tools	[185] (1997)	CP/PARAFAC decomposition	An overview of CP decomposition with respect to aspects: features, properties, methods, and applications in chemometrics.
	[186] (2008)	CP & Tucker decomposition	A literature survey on unsupervised multiway data analysis: multiway models (i.e., CP family and Tucker family), their workhorse algorithms and applications.
	[10] (2009)	CP & Tucker decomposition	An extensive survey on main algorithms, properties and applications of CP, Tucker decompositions and their variants. A list of software and toolboxes for tensor processing.
	[187] (2010)	Tucker/HOSVD decomposition	An overview on numerical methods for Tucker/HOSVD decomposition & its applications in signal processing.
	[188] (2013)	Low-rank tensor approximations	A literature survey on low-rank tensor approximation models and algorithms.
	[189] (2014)	Incomplete tensor decomposition	A survey on numerical methods for factorizing incomplete tensors and their connections to signal processing applications.
	[12] (2016)	Tensor network decomposition	An extensive tutorial on tensor networks, their operations and algorithms.
	[190] (2016)	Big tensor decomposition	A brief review of methods for factorizing large-scale tensors.
	[191] (2020)	Tucker/HOSVD decomposition	A survey on randomized algorithms for computing Tucker/HOSVD decomposition.
	[192] (2020)	Structured tensor decomposition	A unified nonconvex optimization perspective for computing large-scale matrix and tensor decompositions with structured factors.
Surveys on general tensor problems	[193] (2007)	Tensor filtering	A review of tensor signal algebraic filtering methods.
	[194] (2009)	CP & Tucker decompositions	A review of theoretical results on the existence, uniqueness, degeneracies, and numerical complexities of alternating least-squares and other tales.
	[195] (2013)	Complexity of tensor problems	An in-depth survey on theoretical and complexity results of some tensor problems: tensor rank, eigen/singular values, and the best rank-1 approximation.
	[196] (2014)	Tensor formats & tensor ranks	A brief introduction on different types of tensor formats and tensor ranks.
	[11] (2017)	Fundamentals & backgrounds	An comprehensive overview of tensor decompositions w.r.t. aspects: uniqueness, tensor ranks, algorithms, bounds, and applications. A list of software and toolboxes for tensor processing.
	[197] (2018)	Connections to PCA	An introduction to tensors and tensor decompositions in the lens of PCA.
Surveys on tensor applications	[198] (2011)	Data analysis	An overview of tensor decomposition applications for a wide variety of data and problem domains.
	[199] (2015)	Signal processing	A comprehensive survey on tensor decompositions for signal processing.
	[180] (2015)	EEG applications	A brief survey on tensor decompositions of EEG signals.
	[200] (2016)	Anomaly detection	An interdisciplinary survey on tensor-based anomaly detection.
	[201] (2017)	Data fusion	A review of tensor decompositions with emphasis on data fusion applications.
	[202] (2017)	Machine learning & data analysis	An tutorial on tensor network models for super-compressed representation of data and their applications in machine learning and data analytics.
	[203] (2019)	Machine learning	An overview of tensor techniques and applications in machine learning.
	[204] (2021)	Multisensor signal processing	A comprehensive survey on tensor methods for multisensor signal processing.
	[182] (2021)	Wireless communications	A comprehensive overview of tensor decompositions for wireless communications.
	[184] (2021)	Social networks	A survey on tensor decomposition for analysing time-evolving social networks.
	[205] (2021)	Computer vision & deep learning	A practical overview of tensor methods for computer vision and deep learning
	[206] (2022)	Nonlinear system identification	A tutorial on tensor methods for nonlinear system identification.
	[13] (2022)	Data analysis	A systematic and up-to-date overview of tensor decompositions from the engineer's point of view.
	This work	Streaming tensor decomposition (Tensor tracking)	A contemporary and comprehensive survey on methods for factorizing tensors derived from data streams under several tensors formats. Research challenges, open problems, and future directions.

general tensor problems, e.g., tensor operations, uniqueness, ranks, filtering, spectral analysis, and complexity, and (iii) surveys on tensor applications. We refer the readers to Tab. 5.1 for the main contributions of the state-of-the-art surveys on tensors.

Among them, the most notable and highly-cited survey paper is the work of Kolda *et al.* in [10] that was published in the *SIAM Review* journal more than a decade ago. The survey presented basic multiway models (i.e., CP family and Tucker family) and workhorse algorithms for factorizing tensors under these models. Some applications and software for tensors were also mentioned. The second key survey in the literature is the work of Sidiropoulos *et al.* in [11] that appeared five years ago in the *IEEE Transactions on Signal Processing* journal. To fill some gaps in the existing surveys on CP and Tucker decompositions of that time, the authors provided an in-depth overview of tensors with respect to the following aspects: uniqueness, ranks, bounds, algorithms, and applications. Moreover, an up-to-date list of software and toolboxes for tensor computation was provided therein. To extend beyond the two standard multiway models, Cichocki *et al.* conducted a comprehensive tutorial on tensor networks in [12, 202] that appeared in the *Foundations and Trends in Machine Learning* journal. Particularly, its coverage includes tensor network models, the associated operations and algorithms, and their applications. Besides, it also highlighted connections of tensor networks to dimensionality reduction and large-scale optimization problems. Very recently, Liu *et al.* provided a general overview of tensors from the engineer's point of view in the book *Tensor Computation for Data Analysis* [13]. It covers various aspects of tensor computations and decompositions, from operations and well-known multiway representations to tensor-based data analysis techniques and practical applications.

However, to date, we are not aware of any survey paper specifically reviewing the problem of streaming tensor decomposition or tensor tracking. Therefore, it is of great interest to carry out an overview of this topic to enrich the tensor literature.

5.1.2 Main Contributions

In this chapter, we present a contemporary and comprehensive survey on the state-of-the-art online techniques which are capable of factorizing tensors derived from data streams.

Our survey begins with basic coverage of five common tensor decompositions and their main features. They are CP/PARAFAC, HOSVD/Tucker, BTD, tensor-train, and t-SVD. Two kinds of streaming models are then introduced to represent streaming tensors, including single-aspect and multiaspect. Next, we review four main groups of streaming CP decomposition algorithms: (i) subspace-based, (ii) block-coordinate descent, (iii) Bayesian inference, and (iv) multi-aspect streaming CP decomposition. Under the Tucker format, we categorize currently available single-aspect tensor tracking algorithms into two main classes: online tensor dictionary learning and tensor
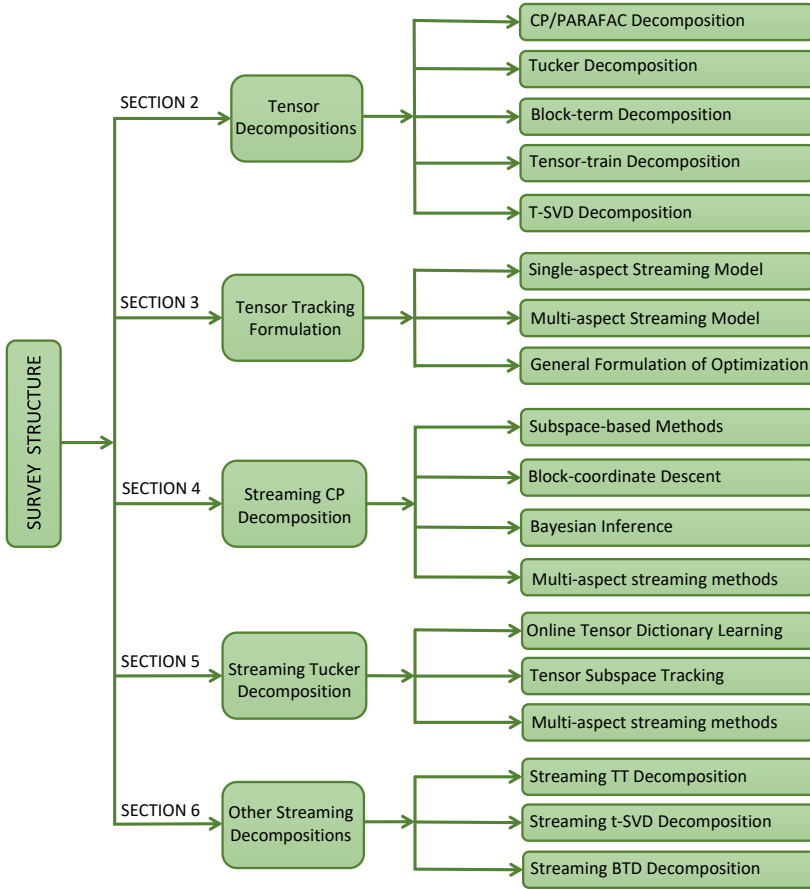
Figure 5.1: Structure of this chapter.

subspace tracking. Multi-aspect streaming Tucker decomposition algorithms are then overviewed. In addition, we provide a short survey on other online techniques for tracking tensors under tensor-train, t-SVD, and BTD formats. Finally, we discuss a number of important challenges and open problems as well as highlight potential directions for the problem of tensor tracking in the future. To the best of our knowledge, our survey offers for the first time a thorough review of techniques for factorizing tensors in an online fashion. Fig. 5.1 depicts depicts the organization of the thesis.

5.2 Tensor Decompositions

In this section, we briefly describe the background of the five popular tensor decompositions which have already been deployed to factorize streaming

tensors in an online fashion. They are CP/PARAFAC, HOSVD/Tucker, BTD, tensor-train, and t-SVD.

5.2.1 CP/PARAFAC Decomposition

Under the CP format [185], a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ can be decomposed into a set of N matrices $\{\mathbf{U}^{(n)}\}_{n=1}^N$ sharing the same number of columns as follows

$$\boldsymbol{\mathcal{X}} \overset{\Delta}{=} [\![\{\mathbf{U}^{(n)}\}_{n=1}^N]\!] = \sum_{i=1}^{r} \mathbf{U}^{(1)}(:,i) \circ \mathbf{U}^{(2)}(:,i) \circ \cdots \circ \mathbf{U}^{(N)}(:,i), \qquad (5.1)$$

where the so-called tensor factor $\mathbf{U}^{(n)}$ is of size $I_n \times r$ with $1 \leq n \leq N$. The smallest r satisfying (5.1) is referred to as the CP-rank of $\boldsymbol{\mathcal{X}}$.

This decomposition has its advantages and disadvantages. On the one hand, CP is the best memory-saving format for representing high-order tensors, and hence, it can overcome the curse of dimensionality which particularly limits the order of tensors to be analysed. Under certain conditions, CP decomposition is essentially unique up to a permutation and scale which is an useful property in many applications, e.g., to recover exact components or individuals hidden in the underlying data. However, its main disadvantage is that finding the true CP-rank r is known as an NP-hard problem [195]. Even though the CP-rank is given in advance, the best rank-r approximation of a tensor may not exist [207]. To compute the CP decomposition, one of the most widely-used approaches is based on the alternating least-squares (ALS) technique [10].

5.2.2 Tucker Decomposition

Under the Tucker format [15], we can factorize $\boldsymbol{\mathcal{X}}$ into a core tensor $\boldsymbol{\mathcal{G}}$ of a smaller size and N factors $\{\mathbf{U}^{(n)}\}_{n=1}^N$ as

$$\boldsymbol{\mathcal{X}} \overset{\Delta}{=} [\![\boldsymbol{\mathcal{G}}; \{\mathbf{U}^{(n)}\}_{n=1}^N]\!] = \boldsymbol{\mathcal{G}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_N \mathbf{U}^{(N)}, \qquad (5.2)$$

where $\boldsymbol{\mathcal{G}}$ is of size $r_1 \times r_2 \times \cdots \times r_N$ with $r_n \leq I_n$, and $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times r_n}$ is an orthogonal matrix. The vector $\mathbf{r} = [r_1, r_2, \ldots, r_N]$ is called the multilinear rank or rank-$(r_1, r_2, \ldots, r_N)$ of $\boldsymbol{\mathcal{X}}$.

Tucker decomposition is more flexible than CP in the sense that we can write any tensor $\boldsymbol{\mathcal{X}}$ in the form (5.2) and its computation can be done effectively and stably. The two most popular algorithms for computing (5.2) are HOSVD and Higher-order Orthogonal Iteration (HOOI) [208]. Both HOSVD and HOOI offer a good rank-$(r_1, r_2, \ldots, r_N)$ tensor approximation for $\boldsymbol{\mathcal{X}}$ and

they can be efficiently implemented in practice. In general, the Tucker representation is not unique but the subspace covering $\mathbf{U}^{(n)}$ is physically unique. Therefore, the main interest in Tucker decomposition is for finding subspaces of the tensor factors, and hence, for approximation, dimensionality reduction, and feature extraction [11].

5.2.3 Block-Term Decomposition

Block-term decomposition (BTD) allows to represent $\mathcal{X}$ as a sum of low multilinear-rank tensors [18]:

$$\mathcal{X} = \sum_{i=1}^{R} [\![\mathcal{G}_i; \{\mathbf{U}_i^{(n)}\}_{n=1}^{N}]\!], \tag{5.3}$$

where $\{\mathcal{G}_r\}_{r=1}^{R}$ with $\mathcal{G}_i \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_N}$ is the set of core tensors, $\mathbf{U}^{(n)} = \left[\mathbf{U}_1^{(n)}, \ldots, \mathbf{U}_R^{(n)} \right]$ with $\mathbf{U}_i^{(n)} \in \mathbb{R}^{I_n \times r_n}$ is the n-th tensor factor, and $r_n \leq I_n \; \forall \, i, n$.

The BTD format (5.3) can be considered as a combination of CP and Tucker. As its name reveals, the basic components in BTD are rank-$(r_1, r_2, \ldots, r_N)$ blocks while they are rank-1 terms in CP/PARAFAC and matrix decompositions. When these blocks are rank-1 tensors (i.e., $r_n = 1 \; \forall n$), it boils down to CP. When it has only one block (i.e., $R = 1$), BTD becomes the standard Tucker decomposition. It is worth noting that the number of blocks R relies on the block's size. Like CP, BTD is essentially unique [18]. The common approach to find (5.3) is also based on the ALS technique [209].

5.2.4 Tensor-train Decomposition

Tensor-train (TT) decomposition expresses $\mathcal{X}$ as a multilinear product of third-order tensors $\{\mathcal{G}^{(n)}\}_{n=1}^{N}$ according to

$$\mathcal{X} = \mathcal{G}^{(1)} \times_1^1 \mathcal{G}^{(2)} \times_2^1 \cdots \times_N^1 \mathcal{G}^{(N)}, \tag{5.4}$$

where $\mathcal{G}^{(n)} \in \mathbb{R}^{r_{n-1} \times I_n \times r_n}$ is the n-th TT-core (aka tensor carriage) with $n = 1, 2, \ldots, N$. Here, $r_0 = r_N = 1$ and the quantities $\{r_n\}_{n=1}^{N-1}$ are called TT-ranks [16].

This type of TD offers several appealing benefits for representing tensors, especially high-order tensors. For instance, given an arbitrary tensor $\mathcal{X}$, we always find a set of TT-cores $\{\mathcal{G}^{(n)}\}_{n=1}^{N}$ satisfying (5.4) with suitable TT ranks. Besides, its TT-ranks can be effectively estimated in a stable way in contrast to the CP-rank determination [195]. Moreover, TT also offers a memory-saving representation for tensors and can break the curse of dimensionality like CP. With respect to the implementation, the workhorse algorithm to compute TT is TT-SVD [16].

5.2.5 T-SVD Decomposition

Tensor SVD (t-SVD) is another multiway extension of SVD for decomposing tensors in which $\mathcal{X}$ is factorized into three tensors $\mathcal{U}$, $\mathcal{G}$, and $\mathcal{V}$ of the same order:

$$\mathcal{X} = \mathcal{U} * \mathcal{G} * \mathcal{V}^H, \tag{5.5}$$

where " $*$ " denotes the t-product, $\mathcal{U}$ and $\mathcal{V}$ are unitary tensors, and $\mathcal{G}$ is a rectangle f-diagonal tensor whose frontal slices are diagonal matrices [17]. To define the low-rank tensor approximation under the t-SVD format, the so-called tubal-rank r_t is determined as the number of non-zero tubes in $\mathcal{G}$, (e.g., when the tensor $\mathcal{X}$ is of order 3, $r_\mathrm{t}(\mathcal{X}) = \sum_i \mathbf{1}[\mathcal{G}(i,i,:) \neq \mathbf{0}]$ where $\mathbf{1}$ is an indicator function).

The t-SVD algebraic framework is quite different from the classical multilinear algebra in other types of TD. Thanks to the t-product and Fourier transform, several linear, multilinear operators and other transformations are successfully extended from matrices to tensors, such as transpose, orthogonality, and inverse. In particular, t-SVD can be effectively obtained by computing SVDs in Fourier domain and its performance (i.e., exact recovery with high probability) can be guaranteed under mild conditions [17].

5.3 Tensor Tracking Formulation

In this section, the problem of tensor tracking is formulated. Specifically, we first divide streaming tensor models into two classes and then construct some terminologies to support the problem statement. They are single-aspect and multi-aspect streaming models, see Fig. 5.2 for an illustration. After that, we formulate a general formulation of the tensor tracking problem which is suitable for many applications.

5.3.1 Single-aspect Streaming Model

In the classical online setting, we are interested in the decomposition of an N-order streaming tensor $\mathcal{X}_t$ fixing all but one dimension (mode). Without loss of generality, we suppose the last dimension is temporal, and hence, we can write $\mathcal{X}_t \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times I_N^t}$ where I_N^t is increasing with time.

The following definition of temporal slices is useful to formulate the problem of single-aspect tensor tracking.
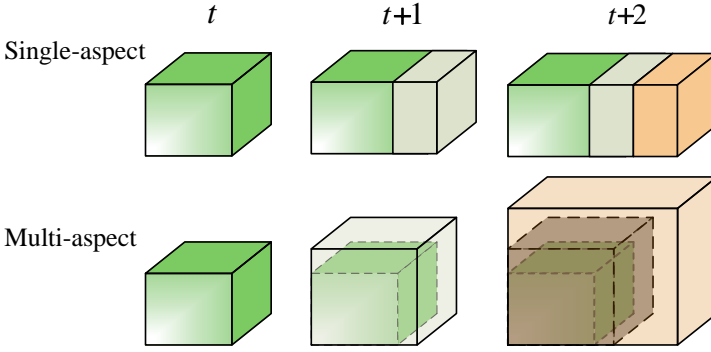
Figure 5.2: Single-aspect and multi-aspect streaming models.

Definition 1 (Temporal slice) *Given a streaming tensor* $\mathcal{X}_t \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times I_N^t}$, *we say* $\mathcal{Y}_\tau = \mathcal{X}_t(:,\ldots,:,\tau) \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1}}$ *is the τ-th temporal slice of* $\mathcal{X}_t$ *for* $1 \le \tau \le I_N^t$.

Without loss of generality, we assume that $I_N^t = t$ meaning that at each time instant one new slice of the tensor is observed. Accordingly, the streaming tensor $\mathcal{X}_t$ can be viewed as a set of temporal slices $\{\mathcal{Y}_\tau\}_{\tau=1}^t$. In other word, $\mathcal{X}_t$ is derived from appending the new comming temporal slice $\mathcal{Y}_t$ to the previous observations $\mathcal{X}_{t-1}$ along the time dimension, i.e.,

$$\mathcal{X}_t = \mathcal{X}_{t-1} \boxplus_N \mathcal{Y}_t \quad \text{and} \quad I_N^t = I_N^{t-1} + 1 = t. \tag{5.6}$$

Generally, $\mathcal{Y}_t$ has the form

$$\mathcal{Y}_t = \mathcal{P}_t \circledast \left(\mathcal{L}_t + \mathcal{N}_t + \mathcal{O}_t \right), \tag{5.7}$$

where " $\circledast$ " denotes the Hadamard product, $\mathcal{L}_t$ is a low-rank component, $\mathcal{P}_t$ is a binary tensor, $\mathcal{N}_t$ is a noise tensor, and $\mathcal{O}_t$ is a sparse tensor. The data model (5.7) is a general form which is suitable for many scenarios. For example, $\mathcal{P}_t$ represents missing and observed entries of $\mathcal{Y}_t$; $\mathcal{N}_t$ is an additive white Gaussian noise; and $\mathcal{O}_t$ denotes the sparse outliers. Meanwhile, the low-rank $\mathcal{L}_t$, which can be formulated by CP, Tucker, BTD, TT, or t-SVD format, can be static or time-varying. Based on these terminologies, the problem of single-aspect tensor tracking can be formally stated as follows:

Single-aspect Tensor Tracking: At time t, given a temporal slice $\mathcal{Y}_t$ and old estimates of $\mathcal{X}_{t-1}$ (e.g., core tensors and tensor factors), we want to track the new estimates of $\mathcal{X}_t = \mathcal{X}_{t-1} \boxplus_N \mathcal{Y}_t$ in time.

5.3.2   Multi-aspect Streaming Model

In some modern online applications, tensor data may evolve in multiple dimensions/modes over time, and thus, the single-aspect streaming model is not useful for modelling such streaming data. In [210], Fanaee-T *et al.* for the first time introduced the concept of multi-aspect streaming tensors to represent streaming data having more than one dimension increasing with time. Since then, some online algorithms have been developed to deal with the problem of multi-aspect streaming tensor decomposition.

For convenience, we first introduce the definitions of multi-aspect streaming tensors and temporal tubes.

> **Definition 2 (Multi-aspect streaming tensor)** *A set of N-order tensors $\{\mathcal{X}_t\}_{t \geq 1}$ is called a multi-aspect streaming tensor sequence denoted as $\{\mathcal{X}\}$ when $\mathcal{X}_t \in \mathbb{R}^{I_1^t \times I_2^t \times \cdots \times I_N^t}$, $I_n^t = I_n^{t-1} + W_n^t$ where $W_n^t \geq 0, 1 \leq n \leq N$, and $\mathcal{X}_{t-1}$ is a sub-tensor of $\mathcal{X}_t$. If $\mathcal{X}_t$ belongs to such a sequence $\{\mathcal{X}\}$, we say that $\mathcal{X}_t$ is a* multi-aspect streaming tensor.

> **Definition 3 (Temporal tube)** *Given two successive tensors $\mathcal{X}_{t-1}$ and $\mathcal{X}_t$ derived from the same multi-aspect streaming tensor sequence $\{\mathcal{X}\}$, the coming data stream at time t can be represented by $\mathcal{Y}_t = \mathcal{X}_t \backslash \mathcal{X}_{t-1}$ of the same size as $\mathcal{X}_t$ with entries*
>
> $$\left[\mathcal{Y}_t\right]_{i_1,\ldots,i_N} = \begin{cases} \left[\mathcal{X}_t\right]_{i_1,\ldots,i_N} & \text{if } I_n^{t-1} < i_n \leq I_n^t, \\ 0 & \text{otherwise,} \end{cases} \quad (5.8)$$
>
> *for $1 \leq n \leq N$. We say that the non-zero entries in $\mathcal{Y}_t$ are* temporal tubes.

Now, we can state the problem of multi-aspect tensor tracking as follows:

> **Multi-aspect Tensor Tracking**: At time t, given temporal tubes in $\mathcal{Y}_t$, and old estimates of $\mathcal{X}_{t-1}$ (e.g., core tensors and tensor factors), we want to track the new estimates of $\mathcal{X}_t = \mathcal{X}_{t-1} \cup \mathcal{Y}_t$ in time.

It is worth noting that the single-aspect tensor tracking problem also belongs to the class of multi-aspect tensor tracking where most of the tensor dimensions I_n are constant by setting $W_n^t = 0$, except the last one I_N^t. Besides, temporal slices may be regarded as frontal slices of the tensor $\mathcal{Y}_t$ defined as in (5.8).

5.3.3   General Formulation of Optimization

We here provide a general formulation of tensor tracking which can be used in many applications. In particular, the optimization problem can be written as

$$
\underset{\{\boldsymbol{\mathcal{G}}\},\{\boldsymbol{\mathcal{U}}\},\boldsymbol{O}}{\text{argmin}} \left[\underbrace{\sum_{\tau=1}^{t} \beta_\tau \ell\Big(\boldsymbol{\mathcal{Y}}_\tau, \boldsymbol{\mathcal{P}}_\tau, \{\boldsymbol{\mathcal{G}}\}, \{\boldsymbol{\mathcal{U}}\}, \boldsymbol{O}\Big)}_{\text{Minimize residual errors}} + \underbrace{\rho_G \mathcal{R}_G(\{\boldsymbol{\mathcal{G}}\})}_{\text{Regularize cores}} + \underbrace{\rho_U \mathcal{R}_U(\{\boldsymbol{\mathcal{U}}\})}_{\text{Regularize factors}} \right.
$$

$$
\left. + \underbrace{\rho_O \mathcal{R}_O(\boldsymbol{O})}_{\text{Promote sparsity}} + \underbrace{\lambda_G \mathcal{L}_G(\{\boldsymbol{\mathcal{G}}\}) + \lambda_U \mathcal{L}_U(\{\boldsymbol{\mathcal{U}}\})}_{\text{Orientate applications}} \right]. \tag{5.9}
$$

Here, $\{\boldsymbol{\mathcal{G}}\}$ and $\{\boldsymbol{\mathcal{U}}\}$ denote the set of core tensors and tensor factors respectively, while $\boldsymbol{O}$ is to represent data corruptions by impulsive noise or outliers. Specifically, the three terms in the second line of (6.5) are used to present regularizations or penalty terms imposed on parameters of interest. The last two penalty terms of (6.5) are for the application orientation. The main loss function $\ell(.)$ is defined to minimize the residual errors between the estimations and observations.

5.4   Streaming CP Decomposition

The primary objective of this section is to provide technical descriptions of the-state-of-the-art online techniques for factorizing streaming tensors under the CP format. In the literature, there are many streaming CP algorithms and they can be categorized into the following groups: (i) subspace-based methods, (ii) block-coordinate descent methods, (iii) Bayesian inference, and (iv) multi-aspect streaming CP decompositions. The three former groups are particularly developed for single-aspect streaming models, while the latter is dedicated to the factorization of tensors having more than one temporally varying mode. The readers are referred to Tabs. 5.2 and 5.3 for quick comparisons of the existing streaming CP decomposition algorithms. In what follows, we take each group into consideration.

5.4.1   Subspace-based Methods

The very first study addressing the problem of streaming CP decomposition is of Nion and Sidiropoulos in [211]. Specifically, the authors introduced the two novel adaptive CP algorithms called PARAFAC-SDT and PARAFAC-RLS capable of tracking third-order streaming tensors having one temporal dimension. Both algorithms are based on the subspace-based approach in which we
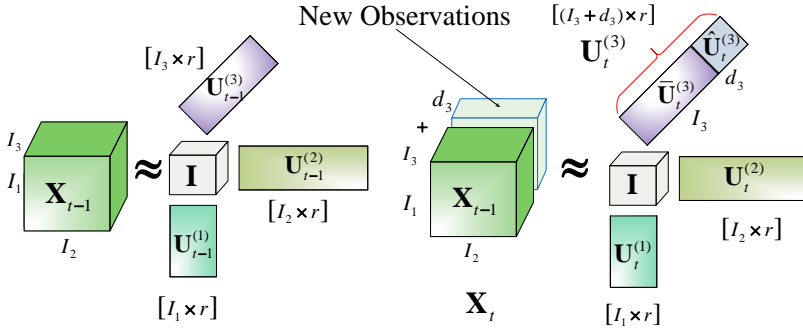
Figure 5.3: Single-aspect streaming CP decomposition of a third-order tensor.

first track a low-dimensional tensor subspace, and then recover the loading matrices from exploiting its Khatri-Rao structure. Following the same line, some other adaptive CP algorithms were proposed for tensor tracking such as CP-PETRELS [215], 3D-OPAST [212], and SOAP [174]. In the following, we describe their subspace-based framework for factorizing streaming tensors with time.

First, we recall that the low-rank $\mathcal{L}_t$ of $\mathcal{Y}_t$ has the form $\mathcal{L}_t = [\![\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1},$ $\mathbf{u}_t^{(N)}]\!]$, where $\mathbf{u}_t^{(N)}$ is the last row of $\mathbf{U}_t^{(N)}$. Thus, $\mathcal{L}_t$ can be recast into the following form:

$$\boldsymbol{\ell}_t \overset{\Delta}{=} \operatorname{vec}(\mathcal{L}_t) = \left[\bigodot_{n=1}^{N-1} \mathbf{U}_t^{(n)} \right] \left(\mathbf{u}_t^{(N)}\right)^\top = \mathbf{H}_t \left(\mathbf{u}_t^{(N)}\right)^\top, \tag{5.10}$$

where $\mathbf{H}_t \in \mathbb{R}^{I_1 \dots I_{N-1} \times r}$ plays a role as a mixing matrix while $\mathbf{u}_t^{(N)}$ can be viewed as a coefficient vector in subspace tracking problems. Accordingly, streaming CP decomposition can boil down to a constrained problem of subspace tracking where the basis matrix has a Khatri-Rao structure.

Particularly for $N = 3$, the authors in [174, 211, 212, 215] proposed to solve the following objective function:

$$\left\{\mathbf{U}_t^{(n)}\right\}_{n=1}^3 = \underset{\{\mathbf{U}^{(n)}\}_{n=1}^3}{\operatorname{argmin}} \sum_{\tau=1}^t \beta^{t-\tau} \left\| \mathbf{p}_\tau \circledast \left(\mathbf{y}_\tau - \mathbf{H}\big(\mathbf{u}_\tau^{(3)}\big)^\top \right) \right\|_2^2$$

$$\text{subject to } \mathbf{H} = \mathbf{U}^{(1)} \odot \mathbf{U}^{(2)}, \tag{5.11}$$

where $\mathbf{y}_\tau = \operatorname{vec}(\mathcal{Y}_\tau)$, $\mathbf{p}_\tau = \operatorname{vec}(\mathcal{P}_\tau)$, and $\mathbf{u}_\tau$ is the τ-th row of the temporal factor $\mathbf{U}_t^{(3)}$, and β is a forgetting factor aimed at discounting the impact of distant observations. Specifically, (5.11) can be effectively solved by applying the following procedure:

Table 5.2: Main features of the state-of-the-art single-aspect streaming CP decomposition algorithms.

Algorithm	Missing Data?	Sparse Outliers?	High-order $(N \geq 4)$?	Convergence Guarantee?	Warm Start?	Computational Complexity	Other Information
PARAFAC-RLST/SDT [211]	✗	✗	✗	✗	✓	$O(r^2I^2)$	- Subspace-based - Tracking using RLST/SDT
3D-OPAST [212]	✗	✗	✗	✗	✓	$O(rI^2)$	- Subspace-based - Tracking using OPAST
TeCPSGD [106]	✓	✗	✗	✗	random	$O(r^2|\Omega|)$	- BCD + SGD
OLCP [175]	✗	✗	✓	✗	✓	$O(r^2I^{N-1})$	- BCD + SGD
SOAP [174]	✗	✗	✗	✗	✓	$O(rI^2)$	- Subspace-based + Second-order estimation - Supports nonnegativity
CP-NLS [213]	✗	✗	✗	✗	✓	$O(r^2I^2)$	- Nonlinear least-squares
BRST [214]	✓	✓	✓	✗	✓	unavailable	- Variational Bayesian
CP-PETRELS [215]	✓	✗	✗	✗	✓	$O(r^2|\Omega|)$	- Subspace-based - Tracking using PETRELS
CP-stream [216]	✗	✗	✓	✗	random	$O(r^2I^{N-1})$	- ADMM + tuning-free - Supports sparsity
POST [217]	✓	✗	✓	✗	✓	$O(r^3NI^{N-1})$	- Variational Bayesian
OLSTEC [176]	✓	✗	✗	✓	random	$O(r^2I^2)$	- BCD + RLS
iPARAFAC [218]	✗	✗	✗	✗	✓	$O(r^2|\mathcal{S}|)$ $|\mathcal{S}|$: size of the selected set	- Apache Spark[a] - Randomized MTTKRP
TensorNOODL [219]	✗	✗	✗	✓	✓	$O(r^2I^2)$	- Online dictionary learning - Supports sparsity
SPADE [220]	✗	✗	✓	✗	✓	$O(r^3I^{N-1})$	- Streaming PARAFAC2[b]
SliceNStitch [221]	✗	✗	✓	✗	random	$O(rN|\mathcal{S}| + (rN)^2 + Nr^3)$ with $|\mathcal{S}|$: number of non-zeros	- Sparse decomposition
SOFIA [222]	✓	✓	✓	✗	✓	$O(r^3I^{N-1})$	- Holt-Winters fitting[c] - Supports seasonality
STF [223]	✓	✗	✓	✗	✓	$O((N+r)Nr|\Omega| + NIr^3)$	- BCD + SGD
ACP [30, 33]	✓	✗	✓	✓	random	$O(r^2|\mathcal{S}|)$ with $|\mathcal{S}|$: size of the selected set	- Random sampling - BCD + RLS
RACP [27]	✓	✓	✓	✓	random	$O(r^2I^{N-1})$	- ADMM + RLS - ℓ_1-norm penalty
Online CPDL [224]	✗	✗	✓	✓	✓	$O(r^2I^{N-1})$	- Nonnegative decomposition - Markovian data - Online dictionary learning

★ Suppose that $I_1 = I_2 = \cdots = I_N = I$, $r_{CP} = r$, and $|\Omega|$ is the number of observed elements.
Abbreviations: RLS (recursive least-squares), SDT (simultaneous diagonalization tracking), BCD (block-coordinate descent), ADMM (alternating direction method of multipliers), SGD (stochastic gradient descent), and MTTKRP (matricized-tensor times Khatri-Rao product).
[a] Apache Spark is a unified data analytics framework that supports distributed storage and large-scale processing: https://spark.apache.org/.
[b] PARAFAC2 is a flexible variant of CP [225]. While the classical CP model requires the tensor factors to be the same for all tensor slices, PARAFAC2 only requires their cross product to be the same and these factors can be different in size slice by slice.
[c] Holt-Winters is an effective time series forecasting procedure [226].

- Stage 1: Estimate $\mathbf{H}_t$ and $\mathbf{u}_t^{(3)}$, given old estimates of $\mathbf{U}_{t-1}^{(1)}$ and $\mathbf{U}_{t-1}^{(2)}$;

- Stage 2: Find $\mathbf{U}_t^{(1)}, \mathbf{U}_t^{(2)}$ satisfying $\mathbf{H}_t \simeq \mathbf{U}_t^{(1)} \odot \mathbf{U}_t^{(2)}$, and then re-update $\mathbf{H}_t \leftarrow \mathbf{U}_t^{(1)} \odot \mathbf{U}_t^{(2)}$;

- Stage 3: Update $\mathbf{U}_t^{(3)} = \left[\left(\mathbf{U}_{t-1}^{(3)}\right)^\top \left(\mathbf{u}_t^{(3)}\right)^\top \right]^\top$ where $\mathbf{u}_t^{(3)}$ can be estimated as in Step 1 (optional).

In stage 1, the authors in [211] proposed two solvers for estimating $\mathbf{H}_t$ and $\mathbf{u}_t$, including recursive least-squares (RLS) and simultaneous diagonalization tracking (SDT). Chinh *et al.* in [215] adopted a well-known subspace tracking algorithm called PETRELS. Dung *et al.* in [212] applied another subspace tracking algorithm for this task, namely OPAST. In [174], the same authors also introduced another low-cost tracker to estimate $\mathbf{H}_t$ with rank-1 updates.

In stage 2, all the existing subspace-based algorithms used the bi-SVD procedure introduced in [227] to recover $\mathbf{U}_t^{(1)}$ and $\mathbf{U}_t^{(2)}$ from $\mathbf{H}_t$. Particularly, we can represent each column of $\mathbf{H}_t$ as $\mathbf{H}_t(:,i) = \mathrm{vec}\big(\mathbf{U}_t^{(1)}(:,i)(\mathbf{U}_t^{(2)}(:,i))^\top\big)$. Accordingly, the right and left singular vector of the reshaped matrix from $\mathbf{H}_t(:,i)$ can provide a good estimate of $\mathbf{U}_t^{(1)}(:,i)$ and $\mathbf{U}_t^{(2)}(:,i)$, respectively,

- $\big[\mathbf{b}_i, \lambda_i, \mathbf{a}_i\big] \leftarrow \mathrm{SVD}\big(\mathrm{reshape}(\mathbf{H}_t(:,i), [I_2\ I_1])\big)$

- $\mathbf{U}_t^{(1)}(:,i) \leftarrow \mathbf{a}_i^*$ and $\mathbf{U}_t^{(2)}(:,i) \leftarrow \lambda_i \mathbf{b}_i$

Computation of SVD may be expensive when dealing with large-scale streaming tensors, we can use the alternative update based on power iteration as follows

- $\mathbf{H}_t^{(i)} \leftarrow \mathrm{reshape}\big(\mathbf{H}_t(:,i), [J \times I]\big)$

- $\mathbf{U}_t^{(1)}(:,i) \leftarrow \big(\mathbf{H}_t^{(i)}\big)^\top \mathbf{U}_{t-1}^{(2)}(:,i)$

- $\mathbf{U}_t^{(2)}(:,i) \leftarrow \dfrac{\mathbf{H}_t^{(i)} \mathbf{U}_t^{(1)}(:,i)}{\big\|\mathbf{H}_t^{(i)} \mathbf{U}_t^{(1)}(:,i)\big\|}.$

As these algorithms are only designed for tracking third-order streaming tensors, there are still rooms to develop subspace-based methods capable of handling $N \geq 4$.

5.4.2 Block-Coordinate Descent

The second approach is based on the block-coordinate descent (BCD) framework in which we decompose the main optimization into two main stages at each time t: (i) estimate the temporal factor $\mathbf{U}_t^{(N)}$ given $\{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}$, and (ii) update the non-temporal factor $\mathbf{U}_t^{(n)}$ with $1 \leq n \leq N-1$ in sequential or parallel given $\mathbf{U}_t^{(N)}$ and the remaining factors. Many tracking algorithms adopt this approach for estimating the low-rank approximation of streaming tensors over time in the literature. We can list here some: TeCPSGD [106], OLCP [175], OLSTEC [176], CP-stream [216], SPADE [220], SOFIA [222], iCP-AM [228], ACP [30], and RACP [27]. In what follows, we review their strategy in each stage.

In stage 1, the general formulation of the optimization to estimate the last row $\mathbf{u}_t^{(N)}$ of $\mathbf{U}_t^{(N)}$ can be given by

$$\{\mathbf{u}_t^{(N)}, \boldsymbol{O}_t\} = \underset{\mathbf{u}^{(N)}, \boldsymbol{O}}{\operatorname{argmin}} \left\| \boldsymbol{\mathcal{P}}_t \circledast \left(\boldsymbol{\mathcal{Y}}_t - \boldsymbol{O} - \left[\!\left[\{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}, \mathbf{u}^{(N)} \right]\!\right] \right) \right\|_F^2$$
$$+ \rho_u \left\| \mathbf{u}^{(N)} \right\|_2^2 + \rho_O \left\| \boldsymbol{O} \right\|_1, \tag{5.12}$$

where $\rho_u \|\mathbf{u}\|_2^2$ is for avoiding the ill-posed computation and $\rho_O \|\boldsymbol{O}\|_1$ promotes the sparsity in $\boldsymbol{O}$. Then, the temporal factor $\mathbf{U}_t^{(N)}$ is obtained by appending the recent updated $\mathbf{u}_t^{(N)}$ to the old estimate $\mathbf{U}_{t-1}^{(N)}$. Most of the existing BCD-based tracking algorithms suppose that observations are outlier-free (i.e., without $\boldsymbol{O}$), and hence, they apply the regularized/randomized least-squares methods for solving (5.12). In the presence of sparse outliers, (5.12) can be effectively minimized by ADMM or shrinkage-thresholding solvers, as presented in SOFIA [222] and RACP [27].

In stage 2, the non-temporal factors $\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}$ can be derived from solving the following optimization

$$\underset{\mathbf{U}^{(n)}}{\operatorname{argmin}} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \underline{\mathbf{P}}_{-\tau}^{(n)} \circledast \left(\mathbf{U}^{(n)} (\mathbf{W}_\tau^{(n)})^\top + \underline{\mathbf{O}}_\tau^{(n)} - \underline{\mathbf{Y}}_\tau^{(n)} \right) \right\|_F^2 + \rho_U \mathcal{R}_U (\mathbf{U}^{(n)}), \tag{5.13}$$

where $\rho_U \mathcal{R}_U(.)$ is a regularization term on $\mathbf{U}^{(n)}$ and

$$\mathbf{W}_\tau^{(n)} = \begin{cases} \left(\underset{i=1, i \neq n}{\overset{N-1}{\odot}} \mathbf{U}_{t-1}^{(i)} \right) \odot \mathbf{u}_\tau^\top & \text{[Jacobi]}, \\ \left(\underset{i=1}{\overset{n-1}{\odot}} \mathbf{U}_t^{(i)} \right) \odot \left(\underset{i=n+1}{\overset{N-1}{\odot}} \mathbf{U}_{t-1}^{(i)} \right) \odot \mathbf{u}_\tau^\top & \text{[Gauss-Seidel]}. \end{cases} \tag{5.14}$$

Here, we can apply one of the two iterative schemes to update $\mathbf{U}_t^{(n)}$: the Jacobi scheme supports the parallel and/or distributed processing while the Gauss-Seidel scheme is useful for a sequential (serial) one. The regularization can be $\|\mathbf{U}^{(n)}\|_F^2$ for smoothness, $\|\mathbf{U}^{(n)} - \mathbf{U}_{t-1}^{(n)}\|_F^2$ for slow time-variation, or $\mathbf{U}^{(n)} \succeq \mathbf{0}$ for non-negativity constraints. Next, we review two common types of solver for optimizing (5.13): adaptive least-squares filters and stochastic gradient solvers.

a) Adaptive Least-Squares (LS) Filters. We can see that the first term of (5.13) is of a weighted LS form very common in adaptive filtering while the second one is to regularize the estimators. Accordingly, (5.13) can be effectively minimized by adaptive LS filters in general and recursive least-squares (RLS) filters in particular.

In [176], Kasai proposed an exponential RLS algorithm called OLSTEC to minimize (5.13) when the observations are outlier-free. OLSTEC is, however, designed for third-order streaming tensors only and its complexities are relatively high compared to other algorithms. Thanh *et al.* in [30] proposed another RLS algorithm called ACP which is capable of dealing with big streaming tensors of higher order ($N \geq 4$). ACP is fast and requires much lower complexity than OLSTEC. Very recently, the same authors in [27] proposed a sliding-window version of ACP robust to both sparse outliers and missing data, namely RACP. Interestingly, three algorithms belong to the class of provable online CP algorithms in which their convergence is guaranteed under certain conditions.

In [213], Vandecappelle *et al.* introduced a nonlinear least-squares (NLS) algorithm for computing the streaming CP decomposition of third-order tensors. In particular, the authors recast the objective function of (5.13) into a truncated exponential window one by incorporating a diagonal weighting matrix $\mathbf{L} = \mathtt{diag}\big([0, \ldots, 0, \beta^{L-1}, \beta^{L-2}, \ldots, \beta, 1]\big)$ and then applied a NLS solver to track the tensor factors with time. Following the same line, Smith *et al.* in [216] proposed another online CP algorithm called CP-stream. This algorithm has the potential to factorize high-order streaming tensors as well as support constraints on streaming CP decomposition such as smoothness and nonnegativity.

b) Stochastic Gradient Solvers. Instead of optimizing (5.13) directly, we can minimize its t-th summand:

$$
\mathbf{U}_t^{(n)} = \underset{\mathbf{U}^{(n)}}{\operatorname{argmin}} \left\| \underline{\mathbf{P}}_t^{(n)} \circledast \left(\underline{\mathbf{Y}}_t^{(n)} - \underline{\mathbf{O}}_t^{(n)} - \mathbf{U}^{(n)} \big(\mathbf{W}_t^{(n)}\big)^\top \right) \right\|_F^2 + \rho_U \mathcal{R}_U \big(\mathbf{U}^{(n)}\big).
$$
(5.15)

Three algorithms TeCPSGD [106], OLCP [175], and SOFIA [222] adopt this replacement for tracking tensor factors with time. The main difference among them is the type of $\mathcal{R}_U(.)$. Besides, they obtain different forms of update:

$$
[\text{SOFIA}] : \ \mathbf{U}_t^{(n)} = \mathbf{U}_{t-1}^{(n)} + \gamma_t \Delta \mathbf{U}_t^{(n)}, \tag{5.16}
$$

$$
[\text{TeCPSGD}] : \ \mathbf{U}_t^{(n)} = \Big(1 - \frac{\beta_t}{t\eta_t}\Big) \mathbf{U}_{t-1}^{(n)} + \frac{1}{\eta_t} \Delta \mathbf{U}_t^{(n)}, \tag{5.17}
$$

$$
[\text{OLCP}] : \ \mathbf{U}_t^{(n)} = \mathbf{P}_t^{(n)} \big(\mathbf{Q}_t^{(n)}\big)^{-1} \text{ with} \tag{5.18}
$$

$$
\mathbf{P}_t^{(n)} = \mathbf{P}_{t-1}^{(n)} + \Delta \mathbf{P}_t^{(n)} \text{ and}
$$

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

Here, $\gamma_t, \eta_t, \Delta \mathbf{U}_t^{(n)}, \Delta \mathbf{P}_t^{(n)}$, and $\Delta \mathbf{Q}_t^{(n)}$ can be obtained from $\big\{\mathbf{U}_{t-1}^{(m)}\big\}_{m=1}^{N-1}$ and the error $\Delta \boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{P}}_t \circledast \big(\boldsymbol{\mathcal{Y}}_t - \boldsymbol{\mathcal{O}}_t - [\![\{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}, \mathbf{u}_t^{(N)}]\!] \big)$. It is worth noting

that SOFIA is capable of dealing with sparse corruptions. TeCPSGD has the ability to track tensors from missing observations, while OLCP can handle streaming tensors of order greater than 3.

In [228], Zeng *et al.* proposed an incremental ALS algorithm called iCP-AM to minimize a reinforced version of (5.15) which is defined as

$$
\mathbf{U}_t^{(n)} = \underset{\mathbf{U}^{(n)}}{\arg\min} \left\| \left[\underline{\mathbf{Y}}_t^{(n)} \quad \mathbf{U}_{t-1}^{(n)} \left(\mathbf{U}_{t-1}^{(N)} \odot \mathbf{V}_{t-1}^{(n)} \right)^\top \right] - \mathbf{U}^{(n)} \left(\begin{bmatrix} \mathbf{u}_t^{(N)} \\ \bar{\mathbf{U}}_t^{(n)} \end{bmatrix} \odot \mathbf{V}_t^{(n)} \right)^\top \right\|_F^2 ,
$$

$$(5.19)$$

where $\mathbf{V}_\tau^{(n)} = \left(\bigodot_{i=1}^{n-1} \mathbf{U}_\tau^{(i)} \right) \odot \left(\bigodot_{i=n+1}^{N-1} \mathbf{U}_\tau^{(i)} \right)$. An appealing feature of iCP-AM against other online CP algorithms is that it has a strategy to deal with the variation of the CP rank over time, i.e., to change the number of low-rank components throughout the tracking process.

In parallel, Gujral *et al.* in [220] proposed an online algorithm called SPADE for tracking tensors under the PARAFAC2 format. Specifically, SPADE tracks a fixed (non-temporal) factor along one mode and allows the other tensor factors (modes) to vary with time. Thanks to its stochastic design, SPADE is fast and memory-efficient. However, the stationary assumption that time variation or concept drift is not allowed limits its applicability.

5.4.3 Bayesian Inference

Besides, another good approach for dealing with the problem of streaming CP decomposition is Bayesian inference. The state-of-the-art Bayesian-based streaming CP decomposition algorithms are POST [217], BRST [214], and SBDT [229]. In general, three algorithms start with a prior distribution of unknown parameters and then infer a posterior that best approximates the joint distribution of these parameters on the arrival of new streaming data. The estimated posterior is then used as the prior for the next update. In this subsection, we briefly describe the two online Bayesian inference frameworks which were already used for tensor tracking: (i) streaming variational Bayes (SVB) and (ii) assumed-density filtering (ADF). Also, prior distributions of parameters of interest are reviewed.

a) Streaming variational Bayes. The two former algorithms POST and BRST adopted the SVB framework [230] which is based on the following Bayes' rule:

$$
p\big(\Theta \big| \mathcal{X}_{t-1} \boxplus_N \mathcal{Y}_t\big) \propto p\big(\mathcal{Y}_t \big| \Theta\big) p\big(\Theta \big| \mathcal{X}_{t-1}\big), \tag{5.20}
$$

where Θ denotes the parameters of interest, e.g., tensor factors, CP rank, noise factors, and other parameters. On the arrival of $\mathcal{Y}_t$, SVB first uses the current

posterior $q_{t-1}(\Theta) := p(\Theta|\mathcal{X}_{t-1})$ as the prior of Θ, and then integrates with the likelihood of $\mathcal{Y}_t$ to obtain

$$\tilde{p}_t(\Theta) = p(\mathcal{Y}_t|\Theta) q_{t-1}(\Theta), \tag{5.21}$$

which can be served as an approximation of the joint distribution $p(\Theta, \mathcal{Y}_t)$ up to a scale factor. The variational posterior $q_t(\Theta)$ is derived from maximizing the variational model evidence lower bound (ELBO)

$$\mathcal{L}(q(\Theta)) = \mathbb{E}_q\left[\log\left(\tilde{p}_t(\Theta)/q(\Theta) \right) \right] \tag{5.22}$$

which is equivalent to minimizing the Kullback-Leibler (KL) divergence:

$$\underset{q}{\text{argmin}} \left[\text{KL}\left(q(\Theta) \| \tilde{p}_t(\Theta) \right) = \int q(\Theta) \log\left\{ \frac{q(\Theta)}{\tilde{p}_t(\Theta)} \right\} d\Theta \right]. \tag{5.23}$$

The optimized form of $q_t(\Theta_i)$ of (5.23) can be given by

$$\log q_t(\Theta_i) = \mathbb{E}_{q(\Theta/\Theta_i)}\left[\log \tilde{p}_t(\Theta) \right] + \text{const}, \tag{5.24}$$

where $\mathbb{E}_{q(\Theta/\Theta_i)}[.]$ is an expectation w.r.t. q over all but Θ_i.

b) Assumed-Density Filtering. The latter algorithm, SBDT, applied the ADF framework to infer the posterior distribution $q_t(\Theta)$ over time. Particularly, ADF is an incremental learning framework that allows for computing the approximate posteriors in Bayesian inference for stochastic processes [231]. The ADF framework is also grounded on the Bayes' rule (5.20) but utilizes a distribution from the exponential family (e.g., Gaussian distribution) to approximate the current posterior. Instead of minimizing the KL divergence or maximizing the variational ELBO like SVB, ADF projects $\tilde{p}_t(\Theta)$ into the selected distribution through moment matching to obtain $q_t(\Theta)$.

c) Prior distributions over Θ. We list common prior distributions over Θ which were already used by POST, BRST, and SBDT.

Prior distribution of tensor factors: All three algorithms assume that the prior over tensor factors is derived from the following Gaussian distribution which is controlled by the hyperparameter $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_r]$:

$$p(\mathbf{U}^{(n)}|\lambda) = \prod_{i=1}^{I_n} \mathcal{N}(\mathbf{u}_i^{(n)}|\mathbf{0}, \Lambda^{-1}), \forall n \in [1, N], \tag{5.25}$$

where $\mathbf{u}_i^{(n)}$ is the i-th row of $\mathbf{U}^{(n)}$ and $\Lambda = \texttt{diag}(\lambda)$ denotes the inverse covariance matrix. Here, λ is supposed to follow a Gamma distribution:

$$p(\lambda) = \prod_{j=1}^{r} \text{Gam}\left(\lambda_j|c_j, d_j \right), \tag{5.26}$$

where $\text{Gam}\left(\lambda_j|c_j, d_j\right) = \frac{d_j{}^{c_j}}{\Gamma(c_j)}\lambda_j{}^{c_j-1}e^{-d_j\lambda_j}$ with $\Gamma(z) = \int_0^\infty x^{z-1}e^{-x}dx$. Specifically, the mean and variance of $\text{Gam}(\lambda_j|c_j, d_j)$ are, respectively, c_j/d_j and c_j/d_j^2 which aim to control the magnitude of $\boldsymbol{\lambda}$.

Prior distribution of noises: The noise tensor is often assumed to be Gaussian, i.e., $\boldsymbol{N}_t \sim \prod_{i_1i_2\ldots i_N} \mathcal{N}(0, \tau^{-1})$ with a noise precision $\tau > 0$. The parameter τ is further assigned to another Gamma distribution $p(\tau|a, b) = \text{Gam}\left(\tau|a, b\right)$ in the same way as for $\boldsymbol{\lambda}$.

Prior distribution of sparse components: Only BRST in [214] has the ability to handle sparse outliers. Here, BRST places a Gaussian prior distribution over the sparse $\boldsymbol{O}_t$ as

$$p(\boldsymbol{O}_t|\boldsymbol{\gamma}) = \prod_{i_1}^{I_1}\prod_{i_2}^{I_2}\cdots\prod_{i_N}^{I_N} \mathcal{N}\left(\left[\boldsymbol{O}_t\right]_{i_1i_2\ldots i_N}\bigg|0, \gamma_{i_1i_2\ldots i_N}^{-1}\right)^{[\mathcal{P}_t]_{i_1i_2\ldots i_N}}. \tag{5.27}$$

where $\boldsymbol{\gamma}$ is the sparsity precision parameter. If the value of $\gamma_{i_1\ldots i_N}$ is large, the corresponding entry in $\boldsymbol{O}_t$ is likely to have a small magnitude. By controlling the value of $\gamma_{i_1\ldots i_N}$, we can control the sparsity of $\boldsymbol{O}_t$.

Prior distribution of NN's weights: SBDT in [229] incorporates neural networks (NN) into tensor factorization. SBDT assigns a spike-and-slab prior distribution over NN weights to sparsify the network. Each weight $\omega_{mjt} = [\mathbf{W}_m]_{jt}$ of NN is particularly sampled from

$$p\left(\omega_{mjt}|s_{mjt}\right) = s_{mjt}\mathcal{N}\left(\omega_{mjt}|0, \sigma_0^2\right) + (1 - s_{mjt})\delta(\omega_{mjt}), \tag{5.28}$$

where $\delta(.)$ denotes the delta function and the binary selection indicator s_{mjt} is derived from $p(s_{mjt}) = \text{Bern}(s_{mjt}|\rho_0) = \rho_0^{s_{mjt}}(1 - \rho_0)^{1-s_{mjt}}$.

5.4.4 Multi-aspect streaming CP decomposition

In the literature, there are some online algorithms capable of tracking multi-aspect streaming tensors under the CP format, such as MAST [232], OR-MSTC [233], InParTen2 [234], and DisMASTD [235]. We refer the readers to Tab. 5.3 for their key features. In what follows, we first describe the main dynamic tensor decomposition (DTD) framework shared by most of these algorithms and then highlight their characteristics in the following text. For ease of reference, we denote by $\boldsymbol{X}_{t-1} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ and $\boldsymbol{X}_t \in \mathbb{R}^{(I_1+d_1)\times\cdots\times(I_N+d_N)}$ the two successive snapshots at $t-1$ and t, please see Fig. 5.4 for an illustration. At time t, given $\boldsymbol{X}_t$ and the old estimates $\{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^N$ of $\boldsymbol{X}_{t-1}$, we wish to update $\{\mathbf{U}_t^{(n)}\}_{n=1}^N$ such that $\boldsymbol{X}_t \approx [\![\{\mathbf{U}_t^{(n)}\}_{n=1}^N]\!]$.

The DTD introduced in [232] offers an online framework for the problem of multi-aspect streaming CP decomposition. Particularly, DTD relaxes the CP representation of $\boldsymbol{X}_t$ in the sense that if $\boldsymbol{X}_t$ is expressed by $[\![\{\mathbf{U}_t^{(n)}\}_{n=1}^N]\!]$,
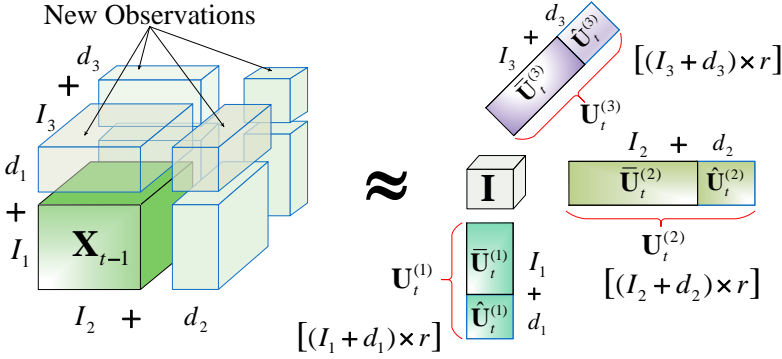
Figure 5.4: Multi-aspect streaming CP decomposition of a third-order tensor.

then its sub-tensor $\mathcal{X}_{t-1}$ can be approximated by $[\![\{\bar{\mathbf{U}}_t^{(n)}\}_{n=1}^N]\!]$ where $\bar{\mathbf{U}}_t^{(n)} \in \mathbb{R}^{I_n \times r}$ is the sub-matrix of $\mathbf{U}_t^{(n)} \in \mathbb{R}^{(I_n+d) \times r}$. Accordingly, DTD enables us to divide $\mathcal{X}_t$ into two parts $\mathcal{X}_{t-1}$ and $\mathcal{Y}_t = \mathcal{X}_t \backslash \mathcal{X}_{t-1}$ in order to take advantages of old estimates. We can first update $\bar{\mathbf{U}}_t^{(n)}$ incrementally from $\mathbf{U}_{t-1}^{(n)}$ with a low cost and then estimate the remaining part $\hat{\mathbf{U}}_t^{(n)} \in \mathbb{R}^{d \times r}$ of $\mathbf{U}_t^{(n)}$. The tensor factors are particularly derived from

$$\{\mathbf{U}_t^{(n)}\}_{n=1}^N = \underset{\{\mathbf{U}^{(n)}\}_{n=1}^N}{\text{argmin}} \, \ell\Big(\mathcal{Y}_t, \{\mathbf{U}^{(n)}\}_{n=1}^N\Big) + \rho\Big(\sum_{n=1}^N \big\|\mathbf{U}^{(n)}\big\|_*\Big), \qquad (5.29)$$

where the loss function $\ell(.)$ is defined as

$$\ell\Big(\mathcal{Y}_t, \{\mathbf{U}^{(n)}\}_{n=1}^N\Big) = \mu \left\| [\![\{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^N]\!] - [\![\{\bar{\mathbf{U}}^{(n)}\}_{n=1}^N]\!] \right\|_F^2$$
$$+ \left\| \mathcal{P}_{\Omega_t}(\mathcal{Y}_t) - \mathcal{P}_{\Omega_t}\Big([\![\{\mathbf{U}^{(n)}\}_{n=1}^N]\!]\Big) \right\|_F^2. \qquad (5.30)$$

Here, Ω_t denotes the set of observed entries and $\mu, \rho > 0$ are two regularized parameters. Depending on the type of constraints, additional information imposed and the method of optimization, we can obtain several types of estimators for tracking multi-aspect streaming tensors with time under the DTD framework.

In [232], Song *et al.* developed the so-called MAST algorithm for tracking multi-aspect streaming tensors. The authors recast (5.29) into a constrained minimization and then formed the following Lagrangian function

$$\mathcal{L}(\mathcal{Y}_t, \Theta) = \sum_{n=1}^N \Big(\rho\big\|\mathbf{Z}^{(n)}\big\|_* + \big\langle \Lambda^{(n)}, \mathbf{Z}^{(n)} - \mathbf{U}^{(n)}\big\rangle + \frac{\eta}{2}\big\|\mathbf{Z}^{(n)} - \mathbf{U}^{(n)}\big\|_F^2 \Big)$$
$$+ \ell\Big(\mathcal{Y}_t, \{\mathbf{U}^{(n)}\}_{n=1}^N\Big), \qquad (5.31)$$

Table 5.3: Main features of multi-aspect streaming CP decomposition algorithms.

Algorithm	MAST (2017 [232])	OR-MSTC (2019 [233])	InParTen2 (2020 [234])	DisMASTD (2021 [235])
Missing?	✓	✓	✗	✗
Outliers?	✗	✓	✗	✗
High-order? $(N \geq 4)$	✓	✓	✗	✓
Distributed?	✗	✗	✓	✓

where $\boldsymbol{\Theta} = \{\mathbf{U}^{(n)}, \mathbf{Z}^{(n)}, \boldsymbol{\Lambda}^{(n)}\}_{n=1}^{N}$ with auxiliary matrices $\{\mathbf{Z}^{(n)}\}_{n=1}^{N}$ and Lagrange multiplier matrices $\{\boldsymbol{\Lambda}^{(n)}\}_{n=1}^{N}$, and $\eta > 0$ is a regularization parameter. Since terms of (5.31) are all convex, it can be effectively minimized by several methods. In particular, MAST applies an ADMM solver to minimize (5.31) in order to balance the trade-off between effectiveness and efficiency in tracking process.

Since MAST is not designed for handling sparse outliers, Najafi *et al.* in [233] introduced a robust version of MAST called OR-MSTC. In the presence of sparse outliers, the authors proposed to regularize the objective function of (5.29) by adding an ℓ_1-norm regularization term $\lambda \|\boldsymbol{O}\|_1$ and replacing $\boldsymbol{\mathcal{Y}}_t$ with $\boldsymbol{\mathcal{Y}}_t - \boldsymbol{O}$ in the first term of $\ell(.)$ in (5.31). Because the term $\lambda \|\boldsymbol{O}\|_1$ is convex, OR-MSTC also adopts the well-known ADMM method in a similar way to MAST.

In [234], Yang *et al.* proposed a distributed version of MAST called InParTen2. Thanks to Apache Spark, it can handle large-scale streaming tensors efficiently with a limited memory. However, the use of InParTen2 is limited for third-order streaming tensors only. In [235], Yang *et al.* introduced another distributed method called DisMASTD capable of dealing with tensors of higher order. One of appealing feature of DisMASTD is that it can avoid repetitive computation and reduce network communication cost.

5.5   Streaming Tucker Decomposition

In the literature, there are many online tensor methods proposed for factorizing streaming tensors. We can broadly categorize them into three main classes: (i) online tensor dictionary learning, (ii) tensor subspace tracking, and (iii) multi-aspect streaming Tucker decomposition. Specifically, the first two classes are designed for two specific cases of single-aspect streaming Tucker decompositions, while the latter class is for multi-aspect streaming tensors.

Table 5.4: Main features of the state-of-the-art streaming Tucker decomposition algorithms.

Algorithm	Missing Data?	Sparse Outliers?	High-order ($N \geq 4$)?	Convergence Guarantee?	Computational Complexity	Additional Information
STA [236, 237]	✗	✗	✓	✗	$O\big((N-1)rI^{N-1}\big)$	- Subspace tracking + deflation
IRTSA [238, 239]	✗	✗	✗	✗	$O\big(3rI^3\big)$ (with $N=3$)	- ISVD-based tracking
ITF [240]	✗	✗	✗	✗	$O\big(3rI^3\big)$ (with $N=3$)	- ISVD-based tracking
IHOSVD [241]	✗	✗	✓	✗	$O\big(Nr^2I^N\big)$	- Adopts recursive matrix SVD
ALTO [242]	✗	✗	✗	✓	$O\big(3(r+k)^6I^3\big)$ k: random columns	- Adds noise perturbation - Uses tensor sequential mapping
LRUT [243]	✗	✗	✓	✗	$O\big(N(r+k)^{2N}I^N\big)$ k: random columns	- Adds noise perturbation - Supports parallel computing
Riemannian-Tucker [244]	✓	✗	✗	✗	unavailable	- Computes SGD on Riemannian manifold
HO-RLSL [245]	✗	✓	✓	✗	$3I^2O(I^3)$	- For $N=4$ only
IHOSVD [246]	✗	✗	✓	✗	$O\big(N(I/d)^{2(N-1)}\big)$ d: number of cores	- Supports distributed computing - Adopts RoundRobin process + columnwise Jacobi-rotation
MIHOSVD [247]	✗	✗	✓	✗	$O\big(N(I/d)^{2(N-1)}\big)$ d: number of cores	- Supports distributed computing - Adopts tree-based integration + columnwise Jacobi-rotation
SIITA [248]	✓	✗	✓	✗	$O\big(K(r^N\lvert\Omega\rvert + NIMr)\big)$ K: iterations, M: number of columns of side matrices	- Multi-aspect streaming method - Supports side information + nonnegativity + sparsity
eOTD [249]	✗	✗	✓	✗	$O\big(rd^{2(m-1)}I^{2(N-m)}\big)$ d, m: number of coming temporal slices & modes	- Multi-aspect streaming method - Adopts SGD + MGS + block tensor matrix multiplications
OTL [250]	✗	✗	✓	✓	$O\big(d(N-1)(Ir^2)^{N-1}\big)$ d: dimensionality of new coming tensor	- Promotes sparse coding - Supports nonnegativity + orthogonality
Singleshot [251]	✗	✗	✓	✓	$O\big(dNr^NI^{N-1} + Nr^{2N}\big)$ d: dimensionality of new coming tensor	- Uses tensor sketching - Supports multiple coming temporal slices + nonnegativity
TTMTS [252]	✗	✗	✓	✓	$O\big((Nk+d)I^N\big)$ $d = (s(1-(s/I)^N)/(1-s/I))$ k, s: parameters of projection	- Uses tensor random projection - Supports one/two-pass approximations
SNBTD [253]	✗	✗	✓	✓	$O\big(I^{N-1}(NIr + MR + 4M^2)\big)$ M: number of pseudo inputs[a] R: size of the pseudo input	- Nonlinear decomposition with Fourier features - Uses Bayesian inference + ADF
D-L1-Tucker [254]	✗	✓	✓	✗	$O\big(K(rI^{N-1} + I^2r^{N-1})\big)$ K: iterations	- Applies threshold-based outlier detection + L1-HOOI
BASS-Tucker [255]	✗	✗	✓	✗	$O\big(r^{3(N-1)} + (Ir)^{N-1} + Nr^3I^{N-1}\big)$	- Sparse decomposition - Uses Bayesian inference + ADF
SBDT [229]	✗	✗	✓	✗	$O\big(NIr + KI^{N-1}\big)$ K: number of weights in NNs	- Uses Bayesian inference + ADF - Incorporates NNs
Zoom-Tucker [256]	✗	✗	✓	✗	$O\big(KBNrI^{N-1} + KN^2r^{N+1} + KN^2r^2I)\big)$ K: iterations & B: blocks	- Supports multiple coming temporal slices - Requires a preprocessing phase
RI/BK-NTD [257]	✗	✗	✓	✗	$O\big(KN(Ir)^N\big)$ K: iterations	- Nonnegative decomposition - Uses NNLS + BCD
ATD [30]	✓	✗	✓	✓	$O\big(r\lvert\Omega\rvert + r^{2N}\lvert\mathcal{S}_1\rvert + r^2\lvert\mathcal{S}_2\rvert + r^2I^{N-2}\big)$ with $\lvert\mathcal{S}_1\rvert, \lvert\mathcal{S}_2\rvert$: size of sampling sets	- Uses BCD + Sampling - Supports parallel computing

★ Suppose that $I_1 = I_2 = \cdots = I_N = I$, $r_1 = r_2 = \cdots = r_N = r$, and $\lvert\Omega\rvert$ is the number of observed elements.

Abbreviations: ISVD, (incremental SVD), SGD (stochastic gradient descent), MGS (modified Gram-Schmidt process), BCD (block-coordinate descent), ADF (assume-density filtering), NN (neural network), and NNLS (nonnegative constrained least-squares solver).

[a] Pseudo inputs: a small active pseudo set, which is not necessarily required to be a subset of the real data, is introduced to break the dependencies between outputs and hence avoid the explicit computation of the full covariance matrix.
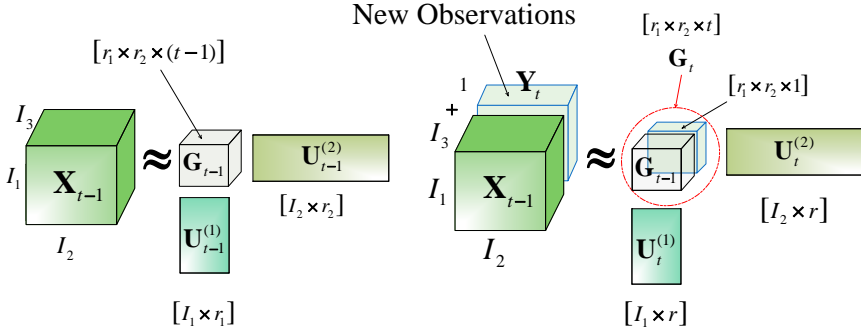
Figure 5.5: Online tensor dictionary learning.

5.5.1 Online Tensor Dictionary Learning

In the class of online tensor dictionary learning methods, we are particularly interested in a specific case of single-aspect streaming Tucker decomposition where the underlying tensor $\boldsymbol{X}_T \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times T}$ – which represents a set of T data streams $\{\boldsymbol{Y}_t\}_{t=1}^T$ of the same size $I_1 \times I_2 \times \cdots \times I_{N-1}$ – is supposed to be modelled by

$$\boldsymbol{X}_T = \left[\!\left[\boldsymbol{G}_T; \{\mathbf{U}^{(n)}\}_{n=1}^{N-1}, \mathbf{I}_T\right]\!\right], \tag{5.32}$$

where the core tensor $\boldsymbol{G}_T$ is of size $r_1 \times \cdots \times r_{N-1} \times T$ (i.e., $r_N = T$), the tensor factors $\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}, \mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times r_n}$ are of fixed size, and the last factor $\mathbf{U}^{(N)}$ is an identify matrix. Specifically, the t-th slice $\boldsymbol{Y}_t$ of $\boldsymbol{X}_T$ is expressed as

$$\boldsymbol{Y}_t = \left[\!\left[\boldsymbol{G}_t; \{\mathbf{U}^{(n)}\}_{n=1}^{N-1}\right]\!\right], \quad t = 1, 2, \ldots, T, \tag{5.33}$$

where $\boldsymbol{G}_t \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_{N-1}}$ is the t-th slice of the core tensor $\boldsymbol{G}_T$. The primary objective here is to estimate $\boldsymbol{G}_t$ and incrementally update $\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}$ on the arrival of $\boldsymbol{Y}_t$ at each time t. In what follows, we review two main approaches to deal with this problem.

a) Incremental Subspace Learning on Tensor Unfolding Matrices. A natural and very first approach for streaming Tucker decomposition is to incrementally update the subspaces covering unfolding matrices of the underlying tensor. The central idea of this approach stems from the fact that the n-th tensor factor $\mathbf{U}_t^{(n)}$ which is derived from the standard HOSVD is given by

$$\mathbf{U}_t^{(n)} = \mathrm{EVD}\left(\left[\underline{\mathbf{X}}_{t-1}^{(n)}, \underline{\mathbf{Y}}_t^{(n)}\right]\left[\underline{\mathbf{X}}_{t-1}^{(n)}, \underline{\mathbf{Y}}_t^{(n)}\right]^\top\right), \tag{5.34}$$

where $\underline{\mathbf{X}}_{t-1}^{(n)} = \left[\underline{\mathbf{Y}}_1^{(n)}, \ldots, \underline{\mathbf{Y}}_{t-1}^{(n)}\right]$ with $\underline{\mathbf{Y}}_\tau^{(n)}$ is the mode-n unfolding matrix of $\boldsymbol{\mathcal{Y}}_\tau$. Accordingly at time t, we can apply the following dynamic tensor analysis (DTA) framework introduced in [236, 237] to estimate $\boldsymbol{\mathcal{G}}_t$ and update $\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}$:

$$\mathbf{C}_t^{(n)} \leftarrow \beta \mathbf{C}_{t-1}^{(n)} + \left(\underline{\mathbf{Y}}_t^{(n)}\right)^\top \underline{\mathbf{Y}}_t^{(n)}, \tag{5.35a}$$

$$\mathbf{U}_t^{(n)} \leftarrow \operatorname{eig}\left(\mathbf{C}_t^{(n)}, r\right), \tag{5.35b}$$

$$\boldsymbol{\mathcal{G}}_t \leftarrow \left[\!\left[\boldsymbol{\mathcal{Y}}_t, \left\{ (\mathbf{U}_t^{(n)})^\top \right\}_{n=1}^{N-1} \right]\!\right], \tag{5.35c}$$

where $0 < \beta \le 1$ is a forgetting factor and $\operatorname{eig}(\mathbf{C}_t^{(n)}, r)$ computes the top r principal eigenvectors of $\mathbf{C}_t^{(n)}$. Since the two steps (5.35a) and (5.35b) are generally expensive, there have been some studies offering good modifications or fast alternatives for (5.35).

In [236, 237], Sun *et al.* proposed a streaming tensor analysis (STA) algorithm for tracking $\mathbf{U}_t^{(n)}$ with time, instead of taking the orthonormal step (5.35b) directly. Particularly on the arrival of $\boldsymbol{\mathcal{Y}}_t$, STA first divides its unfolding matrix $\underline{\mathbf{Y}}_t^{(n)}$ into column vectors $\{\mathbf{y}_{m,t}^{(n)}\}$ and then performs the following steps on each vector $\mathbf{y}_{m,t}^{(n)}$: (i) projects it onto the subspace $\mathbf{U}_{t-1}^{(n)}$, (ii) evaluates the corresponding residual error and the energy for each entry of $\mathbf{y}_{m,t}^{(n)}$, and (iii) updates the matrix $\mathbf{U}_t^{(n)}$. Intuitively, the larger the residual error is, the more $\mathbf{U}_t^{(n)}$ is updated. The complexity of STA is moderate while its effectiveness was demonstrated with the problem of anomaly detection and multi-way latent semantic indexing.

In [238, 239], Hu *et al.* introduced the so-called IRTSA algorithm to track the dominant subspaces $\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}$. Specifically, instead of computing (5.35a), IRTSA applies a fast incremental SVD (ISVD) proposed by Ross *et al.* in [258] on the mode-n unfolding matrix $\underline{\mathbf{X}}_t^{(n)} = \left[\underline{\mathbf{X}}_{t-1}^{(n)}, \underline{\mathbf{Y}}_t^{(n)}\right]$ in (5.34). Thanks to ISVD, IRTSA shares the same order of computational complexity with STA while offers a better estimation than STA for the problem of background modelling and object tracking. Although the current version of IRTSA is designed for factorizing three-order streaming tensors, it is not difficult to extend IRTSA for dealing with higher-order tensors. Besides, a modified version of IRTSA was introduced by Zang *et al.* in [240] for the problem of web service recommendation.

In [241], Kuang *et al.* also proposed an incremental SVD-based streaming Tucker decomposition, namely IHOSVD. In particular, this algorithm performs the following three processes in a serial manner: (i) applies a recursive SVD method to compute singular values and singular vectors of unfolding matrices of the new tensor, (ii) merges the new results with the old estimations from past observations, and (iii) obtains the core tensor with n-mode

products. Theoretical analyses and experimental results on intelligent transportation applications demonstrate the effectiveness of IHOSVD.

In [259], Li *et al.* modified slightly the recursive update of the covariance matrix $\mathbf{C}_t^{(n)}$ in (5.35a) as follows

$$\mathbf{C}_t^{(n)} = (1 - \alpha)\mathbf{C}_{t-1}^{(n)} + \alpha\big(\underline{\mathbf{Y}}_t^{(n)}\big)^\top \underline{\mathbf{Y}}_t^{(n)}, \tag{5.36}$$

with a weight $1 \geq \alpha > 0$ and then introduced a robust incremental algorithm called RTSL which has the potential to model background and detect anomalies in applications of computer vision. Since RTSL still applies directly the DTA framework, its complexity is relatively high. Thus, it may become inefficient for handling large-scale and high dimensional streaming data.

Some other algorithms for streaming Tucker decomposition belonging to this group were presented in [245–247, 260], focusing on specific applications such as dynamic brain network analysis, smart city services, cyber-physical-social networks and systems.

b) Online Multimodal Dictionary Learning. Another good strategy for the problem of single-aspect tensor tracking is to apply online multimodal dictionary learning (OMDL) techniques. As OMDL is a stochastic version of the multimodal dictionary (multilinear subspace) learning [261], it allows estimating dictionaries (i.e., tensor factors) with one-pass processing. In the literature, there exist some algorithms applying OMDL for tracking the low multilinear-rank component of streaming tensors with time, such as OTDL [250], ODL [262], ORLTM [263], OLRTR [264], and D-L1-Tucker [254].

The two former algorithms OTDL and ODL adopt the typical two-step learning procedure to track the tensor factors over time, namely (i) tensor coding or inference of coefficients in the core tensor and (ii) dictionary update per each tensor mode.

Step 1: Tensor Coding. When $\boldsymbol{\mathcal{Y}}_t$ is observed, the general formulation of optimization for this step is given by:

$$\boldsymbol{\mathcal{G}}_t = \underset{\boldsymbol{\mathcal{G}}}{\operatorname{argmin}} \left\| \boldsymbol{\mathcal{Y}}_t - [\![\boldsymbol{\mathcal{G}}; \{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}]\!] \right\|_F^2 + \rho_G \mathcal{R}_G(\boldsymbol{\mathcal{G}}), \tag{5.37}$$

where $\rho_G \mathcal{R}_G(.)$ is a regularization term on the core tensor $\boldsymbol{\mathcal{G}}$ to promote sparsity or nonnegativity for instance. Since the first term of (5.37) is differentiable while the second term may admit a proximal operator (e.g., ℓ_p-norm), OTDL and ODL applied proximal methods to minimize it.

Step 2: Dictionary Update. When $\boldsymbol{\mathcal{G}}_t$ is estimated, the BCD framework can be used to update $\mathbf{U}_t^{(n)}$. Specifically, both algorithms optimize the following minimization:

$$\mathbf{U}_t^{(n)} = \underset{\mathbf{U}^{(n)}}{\operatorname{argmin}} \sum_{\tau=1}^t \left\| \boldsymbol{\mathcal{Y}}_\tau - [\![\boldsymbol{\mathcal{G}}_\tau; \{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}]\!] \right\|_F^2 + \rho_U \mathcal{R}_U\big(\mathbf{U}^{(n)}\big), \tag{5.38}$$

with a penalty term $\rho_U \mathcal{R}_U(.)$ on $\mathbf{U}^{(n)}$. Interestingly, (5.38) can be recast into the standard least-squares cost function which is very common in adaptive filtering theory. Accordingly, OTDL introduced an effective recursive least-squares (RLS) solver to optimize it. Meanwhile, ODL used the stochastic gradient descent method to estimate $\mathbf{U}_t^{(n)}$ with a low cost.

The next two algorithms ORLTM and OLRTR, on the other hand, estimated the tensor factors without the need of tensor coding. In particular, the tensor factor $\mathbf{U}^{(n)}$ is directly derived from the following optimization

$$\mathbf{U}_t^{(n)} = \underset{\mathbf{U}^{(n)}}{\operatorname{argmin}} \sum_{\tau=1}^{t} \ell\big(\boldsymbol{\mathcal{Y}}_\tau, \mathbf{U}^{(n)}\big) + \rho_U \mathcal{R}_U\big(\mathbf{U}^{(n)}\big), \qquad (5.39)$$

where the loss function $\ell(.)$ is defined as

$$\ell\big(\boldsymbol{\mathcal{Y}}_\tau, \mathbf{U}^{(n)}\big) = \min_{\mathbf{R}^{(n)}, \mathbf{E}^{(n)}} \left\| \underline{\mathbf{Y}}_\tau^{(n)} - \mathbf{U}^{(n)}\mathbf{R}^{(n)} - \mathbf{O}^{(n)} \right\|_F^2 + \lambda_1 \big\| \mathbf{E}^{(n)} \big\|_1 + \lambda_2 \mathcal{R}_R(\mathbf{R}^{(n)}).$$
$$(5.40)$$

Here, $\mathbf{R}^{(n)}$ and $\mathbf{O}^{(n)}$ play the role of the coefficient and error, respectively. The main difference between ORLTM and OLRTR is the type of $\mathcal{R}_R(.)$ used. Specifically, OLRTR uses the simple Frobenius norm regularization $\mathcal{R}_R(\mathbf{R}^{(n)}) = \|\mathbf{R}^{(n)}\|_F^2$, while ORLTM reinforces $\mathbf{R}^{(n)} = \mathbf{W}^{(n)}\mathbf{Z}^{(n)}$ and then forms $\mathcal{R}_R(\mathbf{R}^{(n)}) = \|\mathbf{W}^{(n)}\|_F^2 + \|\mathbf{Z}^{(n)}\|_F^2$. Intuitively, the minimization (5.39) may be regarded as a robust version of (5.38) which aims to deal with sparse corruptions. Also, the minimization (5.40) is not difficult to solve since its terms are all convex. Hence, both OLRTR and ORLTM applied the RLS method to update $\mathbf{U}_t^{(n)}$ over time.

In [254], Chachlakis *et al.* proposed a streaming Tucker decomposition called D-L1-Tucker for dealing with streaming tensors. D-L1-Tucker shares the same objective function with ORLTM and OLRTR, but adopts a different approach to handle data corruptions. Particularly on the arrival of $\boldsymbol{\mathcal{Y}}_t$, D-L1-Tucker first identifies whether $\boldsymbol{\mathcal{Y}}_t$ is an anomaly or not based on its reliability which is defined as

$$r_t = \left\| [\![\boldsymbol{\mathcal{Y}}_t; \{(\mathbf{U}_{t-1}^{(n)})^\top\}_{n=1}^{N-1}]\!] \right\|_F^2 \left\| \boldsymbol{\mathcal{Y}}_t \right\|_F^{-2}. \qquad (5.41)$$

If $r_t \leq \tau$ where $\tau \in [0,1]$ is a predefined threshold, $\boldsymbol{\mathcal{Y}}_t$ is labelled as an outlier slice and then it is disregarded. Otherwise, $\boldsymbol{\mathcal{Y}}_t$ is considered as reliable and useful for tracking process. In such a case, D-L1-Tucker appends $\boldsymbol{\mathcal{Y}}_t$ to the memory set $\mathcal{Z}_t = \mathcal{Z}_{t-1} \cup \boldsymbol{\mathcal{Y}}_t$ and then applies the batch L1-HOOI algorithm proposed in [265] for factorizing $\mathcal{Z}_t$ in order to obtain tensor factors. After that, $\mathcal{Z}_t$ is re-updated by removing the oldest measurement for the next processing. D-L1-Tucker requires a good batch initialization and its tracking ability is dependent on the threshold τ and the memory size M to store $\mathcal{Z}_t$.
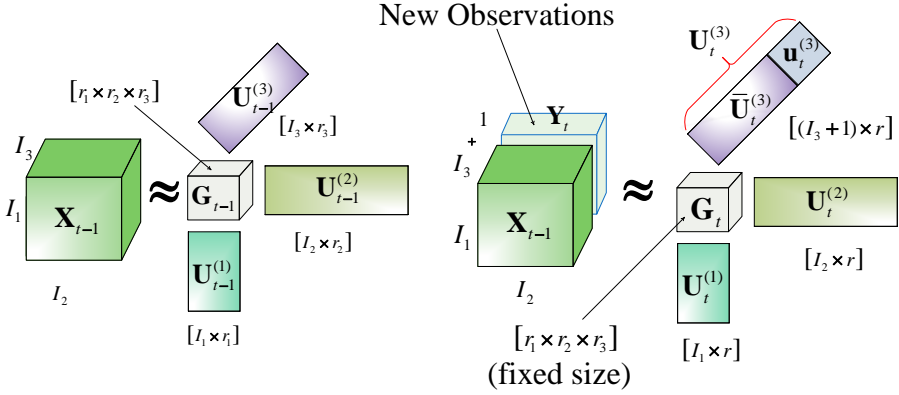
Figure 5.6: Online tensor subspace learning.

Tensor Subspace Tracking

Apart from the model (5.32), the tensor $\boldsymbol{X}_T \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times T}$ and its t-th temporal slice $\boldsymbol{Y}_t$ with $1 \le t \le T$ can be modelled as follows

$$\boldsymbol{X}_T = \left[\!\!\left[\boldsymbol{G}; \left\{ \mathbf{U}^{(n)} \right\}_{n=1}^{N} \right]\!\!\right], \tag{5.42}$$

$$\boldsymbol{Y}_t = \left[\!\!\left[\boldsymbol{G}; \left\{ \mathbf{U}^{(n)} \right\}_{n=1}^{N-1}, \mathbf{u}_t^{(N)} \right]\!\!\right], \tag{5.43}$$

where the core tensor $\boldsymbol{G} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_N}$ and $\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}$ with $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times r_n}$ are of fixed size except the last factor $\mathbf{U}^{(N)} \in \mathbb{R}^{T \times r_N}$, and $\mathbf{u}_t^{(N)} \in \mathbb{R}^{1 \times r_N}$ is the t-th row of $\mathbf{U}^{(N)}$, see Fig. 5.6 for an illustration. At each time t, given old estimations $\boldsymbol{G}_{t-1}$ and $\{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}$, we are interested in tracking $\boldsymbol{G}_t$, $\mathbf{u}_t^{(N)}$ and $\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}$ which can compactly represent the temporal slice $\boldsymbol{Y}_t$. We refer this problem to as tensor subspace tracking.[1]

It is worth mentioning that single-aspect streaming CP methods also belong to this class as the core tensor $\boldsymbol{G}$ is constrained to be identity. In the literature, there exist some tensor subspace tracking methods which have the potential to deal with a general case of $\boldsymbol{G}$. Each method adopts a different strategy to factorize streaming tensors. In what follows, we briefly describe their main features in chronological order.

---

[1]This name stems from the following observation: we can recast (5.43) into the form $\mathbf{y}_t = \mathbf{D}\mathbf{u}_t$, where $\mathbf{y}_t = \mathrm{vec}(\boldsymbol{Y}_t)$, $\mathbf{u}_t = (\mathbf{u}_t^{(N)})^\top$ and $\mathbf{D}$ is the transpose of the mode-N unfolding matrix of $[\![\boldsymbol{G}; \{\mathbf{U}^{(n)}\}_{n=1}^{N-1}]\!]$. Intuitively, this form may be regarded as the data model which is very common and widely used in the problem of subspace tracking where we wish to incrementally update $\mathbf{D}$ on the arrival of $\mathbf{y}_t$ at each time t. Since the subspace matrix $\mathbf{D}$ has a tensor structure, we label this problem as "tensor subspace tracking" without hesitation.

a) Augmented Projection. In [243], Baskaran *et al.* introduced the so-called LRUT algorithm (which stands for Low-Rank Updates to Tucker decomposition) using a randomized projection technique for tracking the low multilinear-rank approximation of streaming tensors over time. When a data stream arrives, LRUT first projects it onto an extended tensor subspace and then forms an augmented core tensor. Specifically, LRUT adds a few more random dimensions to the current tensor subspace defined by old estimations of the tensor factors. The inclusion of some random vectors here plays a role of noise perturbation aimed to prevent the main optimization from getting stuck in local optima. Next, LRUT performs the standard Tucker decomposition (e.g., batch HOSVD or HOOI) on the resulting augmented core tensor to update tensor factors. In this way, we can avoid the computation of SVD on unfolding matrices of the full tensor which is highly expensive in an online setting. However, its computational complexity is still relatively high since LRUT uses several orthogonalization operations on augmented tensor factors and unfolding matrices of the projected tensor slice.

b) Riemannian Optimization. In [244], Kasai *et al.* developed a Riemannian manifold preconditioning approach for tensor completion. Specifically, its stochastic version can be adapted for factorizing incomplete streaming tensors in an online fashion. Since the Tucker format provides an effective representation for tensors in the manifold $\mathcal{M}_{\mathbf{r}} = \{\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} |$ rank$(\mathcal{X}) := \mathbf{r} = [r_1, r_2, \ldots, r_N]\}$, Riemannian optimization can offer a good approach for tensor decomposition and completion [266]. Accordingly, the authors proposed an efficient Riemannian gradient based method to estimate the low multilinear-rank component of tensors. The proposed method consists of a rank-one Riemannian gradient computation and a retraction step. Specifically, a novel Riemannian metric on the tangent space of $\mathcal{M}_{\mathbf{r}}$ and its quotient manifold was introduced to enable the Riemannian optimization framework. Furthermore, a map that combines all retractions on the individual manifolds of tensor factors was used to transform the estimations to the tensor manifold.

c) Bayesian Inference. In [255], Fang *et al.* proposed a Bayesian streaming Tucker decomposition method called BASS-Tucker for handling streaming sparse tensors. Similar to Bayesian methods for streaming CP decomposition, BASS-Tucker adopts the streaming variational Bayes (SVB) framework to infer the posterior of parameters of interest (e.g., tensor core, tensor factors, and nuisance parameters) over time. In addition, BASS-Tucker also utilizes the same priors for the tensor factors and noise variance except that of the core tensor. Here, the following spike-and-slab prior is used to model the

core tensor:

$$p(\mathcal{S}|\rho_0) = \prod_{j_1=1}^{r_1} \prod_{j_2=1}^{r_2} \cdots \prod_{j_N=1}^{r_N} \text{Bern}\left(s_{j_1 j_2 \dots j_N}|\rho_0\right), \tag{5.44}$$

$$p(\mathcal{G}|\mathcal{S}) = \prod_{j_1=1}^{r_1} \prod_{j_2=1}^{r_2} \cdots \prod_{j_N=1}^{r_N} s_{j_1 j_2 \dots j_N} \mathcal{N}\left(g_{j_1 j_2 \dots j_N}|0, \sigma_0^2\right) + (1 - s_{j_1 j_2 \dots j_N})\delta(g_{j_1 j_2 \dots j_N}), \tag{5.45}$$

where $\mathcal{S} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_N}$ is a binary tensor, $\text{Bern}(.|\rho_0)$ is the Bernoulli distribution with probability ρ_0, and $\delta(.)$ is the Delta function. We refer the readers to subsection 5.4.3 for details on prior distributions of $\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}$ and other model parameters as well as how the SVB framework works.

d) Block-Coordinate Descent. There are three online Tucker algorithms using the BCD framework, including ATD [30], RT-NTD [257] and BK-NTD [257]. In general, they go through the following stages when $\mathcal{Y}_t$ arrives:

Stage 1: Estimate the coefficient vector $\mathbf{u}_t^{(N)}$ given old estimations $\mathcal{G}_{t-1}$ and $\{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}$. Generally, $\mathbf{u}_t^{(N)}$ can be derived from

$$\mathbf{u}_t^{(N)} = \underset{\mathbf{u}^{(N)}}{\text{argmin}} \left\| \mathcal{Y}_t - [\![\mathcal{G}_{t-1}; \{\mathbf{U}_{t-1}^{(n)}\}_{n=1}^{N-1}, \mathbf{u}^{(N)}]\!] \right\|_F^2 + \rho_u \mathcal{R}_u(\mathbf{u}^{(N)}). \tag{5.46}$$

Stage 2: Estimate the tensor factor $\mathbf{U}_t^{(n)}$ given $\mathbf{u}_t^{(N)}$, old estimation of $\mathbf{U}_{t-1}^{(n)}$ and the remaining factors, $1 \leq n \leq N-1$. The main optimization can be given by

$$\mathbf{U}_t^{(n)} = \underset{\mathbf{U}^{(n)}}{\text{argmin}} \sum_{\tau=1}^t \beta^{t-\tau} \ell(\mathcal{Y}_\tau, \mathbf{U}^{(n)}) + \rho_U \mathcal{R}_U(\mathbf{U}^{(n)}), \tag{5.47}$$

where $\ell(\mathcal{Y}_\tau, \mathbf{U}^{(n)}) = \left\| \underline{\mathbf{Y}}_\tau^{(n)} - \mathbf{U}^{(n)} \mathbf{W}_\tau^{(n)} \right\|_F^2$, $\underline{\mathbf{Y}}_\tau^{(n)}$ and $\mathbf{W}_\tau^{(n)}$ are respectively the mode-n unfolding matrices of $\mathcal{Y}_\tau$ and $\mathcal{W}_\tau$. Here, the coefficient tensor $\mathcal{W}_\tau$ is defined as

$$\mathcal{W}_\tau = [\![\mathcal{G}_{t-1}; \{\mathbf{U}_{t-1}^{(m)}\}_{m=1, m \neq n}^{N-1}, \mathbf{u}_\tau^{(N)}]\!]. \tag{5.48}$$

Stage 3: Estimate the core tensor $\mathcal{G}_t$ given $\mathcal{G}_{t-1}$, $\mathbf{u}_t^{(N)}$, and $\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}$ particularly from

$$\mathcal{G}_t = \underset{\mathcal{G}}{\text{argmin}} \sum_{\tau=1}^t \beta^{t-\tau} \left\| \underline{\mathbf{X}}_\tau^{(1)} - \mathbf{U}_t^{(1)} \underline{\mathbf{G}}^{(1)} \mathbf{Z}_\tau \right\|_F^2 + \rho_G \mathcal{R}_G(\mathcal{G}), \tag{5.49}$$

where $\underline{(.)}^{(1)}$ denotes the mode-1 unfolding matrix and $\mathbf{Z}_\tau = \mathbf{u}_\tau \otimes \left(\bigotimes_{n=2}^N \mathbf{U}_t^{(n)} \right)$.
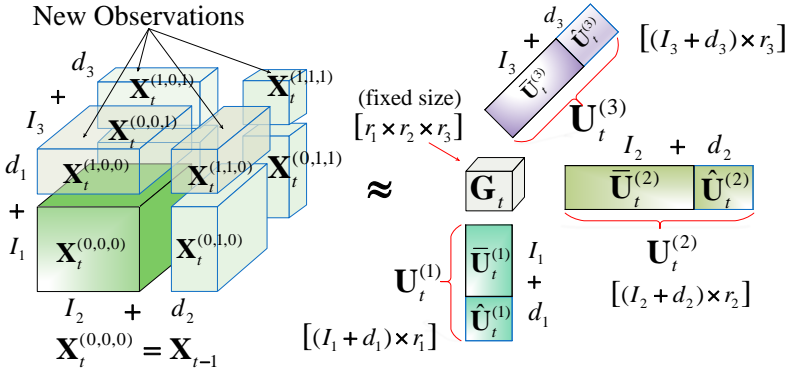
Figure 5.7: Multi-aspect streaming Tucker decomposition of a three-order tensor.

Here, $\mathcal{R}_u(.)$, $\mathcal{R}_U(.)$, and $\mathcal{R}_G(.)$ are regularization terms on the coefficient $\mathbf{u}_t^{(N)}$, the factor $\mathbf{U}_t^{(n)}$, and the core tensor $\mathcal{G}_t$, respectively. These penalties can be nonnegativity, smoothness, or sparsity depending on the specific application.

The former ATD algorithm was proposed by Thanh *et al.* in [30] which is capable of tracking the low multilinear-rank approximation of streaming tensors from highly incomplete observations. In stage 1, ATD particularly recasts (5.46) into a standard LS optimization and then applies a randomized LS technique to minimize it. In stage 2, ATD introduces a recursive LS solver to optimize (5.47) in an efficient way. Instead of solving (5.49) directly, ATD applies the stochastic gradient descent to obtain its solution.

The two latter RI-NTD and BK-NTD algorithms were proposed by Zdunek *et al.* in [257] for factorizing nonnegative tensors from streaming data. Both algorithms perform nonnegative least-square (NNLS) solvers to incrementally update the tensor factors and the core tensor. Particularly, RI-NTD utilizes a recursive strategy involving the nonnegatively constrained Gauss–Seidel method while BK-NTD adopts the block Kaczmarz method. Similar to ATD, both RI-NTD and BK-NTD estimate the core tensor using only the new coming data via a stochastic optimization.

5.5.3   Multi-aspect streaming Tucker decomposition

Besides single-aspect streaming Tucker decomposition methods, few online techniques are capable of tracking multi-aspect streaming tensors under the Tucker format over time, such as SITTA in [248] and eOTD in [249].

SIITA in [248] offers an online inductive framework for tracking the low-rank tensor approximation of multi-aspect streaming tensors as well as com-

pleting their missing data with side information. On the arrival of new data, SIITA particularly minimizes the following optimization

$$\underset{\mathcal{G}, \{\mathbf{U}^{(n)}, \mathbf{A}^{(n)}\}_{n=1}^{N}}{\text{argmin}} \quad f_t\left(\boldsymbol{\mathcal{Y}}_t, \{\mathbf{S}_t^{(n)}\}_{n=1}^{N}, \boldsymbol{\mathcal{G}}, \{\mathbf{U}^{(n)}\}_{n=1}^{N}\right), \tag{5.50}$$

with

$$f_t\left(\boldsymbol{\mathcal{Y}}_t, \{\mathbf{S}^{(n)}\}_{n=1}^{N}, \boldsymbol{\mathcal{G}}, \{\mathbf{U}^{(n)}\}_{n=1}^{N}\right) = \left\|\mathcal{P}_{\Omega_t}\left(\boldsymbol{\mathcal{Y}}_t\right) - \mathcal{P}_{\Omega_t}\left(\llbracket \boldsymbol{\mathcal{G}}; \{\mathbf{S}_t^{(n)}\mathbf{U}^{(n)}\}_{n=1}^{N} \rrbracket\right)\right\|_F^2$$

$$+ \rho_G \left\|\boldsymbol{\mathcal{G}}\right\|_F^2 + \sum_{n=1}^{N} \rho_n \left\|\mathbf{U}^{(n)}\right\|_F^2, \tag{5.51}$$

where $\{\mathbf{S}_t^{(n)}\}_{n=1}^{N}$ with $\mathbf{S}_t \in \mathbb{R}^{M_n \times I_n}$ is the set of side information matrices and $\rho_G, \{\rho_i\}_{i=1}^{N}$ are regularization parameters. Here, SIITA incorporates the side information into the data model by using $\{\mathbf{S}_t^{(n)}\}_{n=1}^{N}$ as multiplicative terms. Accordingly, SIITA can accelerate the tracking process because the product $\mathbf{S}_t^{(n)}\mathbf{U}^{(n)}$ transforms the dimensionality of variables from I_n to M_n, and typically with $M_n \ll I_n$. As every term of (5.50) are convex, SITTA adopts the gradient descent to minimize it. Besides, a simple variant of SIITA namely NN-SITTA was also obtained for nonnegative tensor decomposition. NN-SITTA is specifically derived from projecting the estimates of SIITA into their nonnegative orthant at each time t.

In [249], Xiao *et al.* proposed the so-called eOTD algorithm for the multi-aspect tensor tracking problem. Unlike SIITA, eOTD adopts the divide and conquer paradigm to deal with multi-aspect streaming tensors. In particular, it divides the underlying tensor $\boldsymbol{\mathcal{X}}_t$ into 2^N sub-tensors $\boldsymbol{\mathcal{X}}_t^{(i_1,\dots,i_N)}$ with $i_n \in \{0, 1\}, 1 \leq n \leq N$, and $\boldsymbol{\mathcal{X}}_t^{(0,\dots,0)} = \boldsymbol{\mathcal{X}}_{t-1}$, see Fig. 5.7 for an illustration. These sub-tensors are grouped into N classes $\{\mathcal{X}_n\}_{n=1}^{N}$ based on the sum of sub-indices. For example, for a third-order tensor, we have $\mathcal{X}_1 = \{\boldsymbol{\mathcal{X}}_t^{(1,0,0)}, \boldsymbol{\mathcal{X}}_t^{(0,1,0)}, \boldsymbol{\mathcal{X}}_t^{(0,0,1)}\}, \mathcal{X}_2 = \{\boldsymbol{\mathcal{X}}_t^{(1,1,0)}, \boldsymbol{\mathcal{X}}_t^{(1,0,1)}, \boldsymbol{\mathcal{X}}_t^{(0,1,1)}\}$, and $\mathcal{X}_3 = \{\boldsymbol{\mathcal{X}}_t^{(1,1,1)}\}$. If a sub-tensor $\boldsymbol{\mathcal{X}}_t^{(i_1,\dots,i_N)} \in \mathcal{G}_n$, factorizing it will results in $\boldsymbol{\mathcal{X}}_t^{(i_1,\dots,i_N)} = \llbracket \boldsymbol{\mathcal{G}}_t, \{\mathbf{V}_t^{(n)}\}_{n=1}^{N} \rrbracket$ where $\mathbf{V}_t^{(n)} = \hat{\mathbf{U}}_t^{(n)}$ if $i_n = 1$ and $\mathbf{V}_t^{(n)} = \mathbf{U}_t^{(n)}$ if $i_n = 0$. Here, the matrix $\hat{\mathbf{U}}_t^{(n)}$ is constantly updated as follows

$$\hat{\mathbf{U}}_{new}^{(n)} = \alpha \hat{\mathbf{U}}_{old}^{(n)} + (1 - \alpha) \underline{\mathbf{X}}_{n}^{t\,(i_1,\dots,i_N)} \left(\mathbf{G}_{i_n}^{(n)}\right)^{\#}. \tag{5.52}$$

The tensor factor $\mathbf{U}_t^{(n)}$ is specifically derived from $\mathbf{U}_t^{(n)} = \text{orth}\left([\mathbf{U}_{t-1}^{(n)}; \hat{\mathbf{U}}_{new}^{(n)}]\right) = \left([\bar{\mathbf{U}}_1^{(n)}; \hat{\mathbf{U}}_t^{(n)}]\right)$ where the modified Gram-Schmidt process was applied to compute the orth(.) operation. Finally, the tensor core $\boldsymbol{\mathcal{G}}_t$ of fixed size is estimated
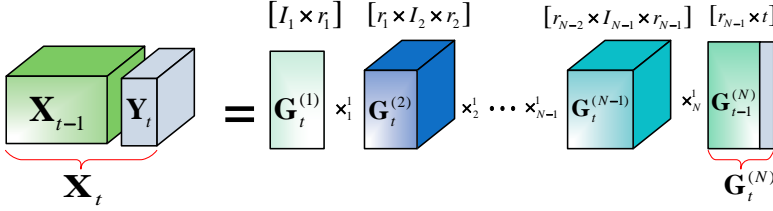
Figure 5.8: Single-aspect streaming tensor-train decomposition.

by

$$
\boldsymbol{\mathcal{G}}_t = \left[\!\!\left[\boldsymbol{\mathcal{G}}_{t-1}, \left\{ \left(\bar{\mathbf{U}}_t^{(n)}\right)^{\top} \mathbf{U}_{t-1}^{(n)} \right\}_{n=1}^{N} \right]\!\!\right] + \sum_{(i_1,\dots,i_N)\neq(0,\dots,0)} \left[\!\!\left[\boldsymbol{\mathcal{X}}_t^{(i_1,\dots,i_N)}, \left\{ \hat{\mathbf{U}}_t^{(n)} \right\}_{n=1}^{N} \right]\!\!\right].
$$

(5.53)

An appealing feature of eOTD is that throughout the tracking process, eOTD only uses cheap tensor-matrix multiplications and pseudo-inverse operations instead of computing the expensive SVDs on big matrices. This makes eOTD easy for applying to large-scale applications.

5.6 Other Streaming Tensor Decompositions

Apart from the two most popular streaming CP and Tucker decompositions, some online methods are capable of tracking tensors under other multiway models. This section focuses on tracking algorithms that exploit TT, BTD, and t-SVD formats to construct the low-rank tensor approximation in the streaming model.

5.6.1 Streaming Tensor-Train Decomposition

Despite success in the batch setting, TT decomposition has not gained in popularity as CP and Tucker for tensor tracking. In the literature, there exist few tracking algorithms developed for the problem of single-aspect tensor tracking under the TT format, see Fig. 5.8 for an illustration.

In [31, 32, 35], Thanh *et al.* proposed three adaptive TT algorithms called TT-FOA, ATT, and ROBOT for factorizing tensors in an online fashion. Particularly, TT-FOA in [32] is, to the best of our knowledge, the very first of its kind in the literature. However, its practical use is limited due to the lack of robustness to data corruption. To overcome the drawback, ATT in [31] and ROBOT in [35] were developed to deal with missing data and sparse outliers, respectively.

All three algorithms share the same optimization framework where block-coordinate gradient (BCD) and recursive least-squares (RLS) methods are utilized to minimize the cost function. In particular, a general formulation of the optimization problems can be written as

$$
\{\{\boldsymbol{\mathcal{G}}_t^{(n)}\}_{n=1}^N, \boldsymbol{O}_t\} = \underset{\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^N, \boldsymbol{O}}{\arg\min} \sum_{\tau=1}^t \beta^{t-\tau} \Bigg(\Big\| \boldsymbol{\mathcal{P}}_\tau \circledast \Big(\boldsymbol{\mathcal{G}}^{(1)} \times_2^1 \cdots \times_{N-1}^1 \boldsymbol{\mathcal{G}}^{(N-1)}
$$

$$
\times_N^1 \mathrm{G}_\tau^{(N)} + \boldsymbol{O}_\tau - \boldsymbol{\mathcal{Y}}_\tau \Big) \Big\|_F^2 + \rho_O \mathcal{R}_O(\boldsymbol{O}_\tau) \Bigg) + \rho_G \mathcal{R}_G\Big(\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^{N-1}\Big), \quad (5.54)
$$

where $\beta \in (0,1]$ is a forgetting factor to reduce the impact of old observations; $\mathcal{R}_O(\boldsymbol{O}_\tau)$ and $\mathcal{R}_G(\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^{N-1})$ are two regularization terms. Specifically, TT-FOA does not impose the two penalties; ATT adopts $\mathcal{R}_G(\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^{N-1}) = \sum_{n=1}^{N-1} \|\boldsymbol{\mathcal{G}}^{(n)} - \boldsymbol{\mathcal{G}}_{t-1}^{(n)}\|_F^2$ to control the smoothness of TT-cores over time; and ROBOT applies the ℓ_1-norm regularization $\mathcal{R}_O(\boldsymbol{O}_\tau) = \|\boldsymbol{O}_\tau\|_1$ to promote the sparsity on $\boldsymbol{O}_\tau$.

Thanks to the BCD framework, (7.46) can be effectively decomposed into two main stages: (i) estimate the temporal TT-core $\mathrm{G}_t^{(N)}$ and outlier $\boldsymbol{O}_t$, and (ii) update non-temporal TT-cores $\{\boldsymbol{\mathcal{G}}_t^{(n)}\}_{n=1}^{N-1}$. In stage 1, TT-FOA and ATT apply the regularized least-squares method to estimate $\mathrm{G}_t^{(N)}$ under the assumption that $\boldsymbol{\mathcal{Y}}_t$ is outlier-free. Meanwhile ROBOT adopts an effective ADMM solver to account for the sparse outlier $\boldsymbol{O}_t$. In stage 2, an effective RLS solver was introduced to estimate $\{\boldsymbol{\mathcal{G}}_t^{(n)}\}_{n=1}^{N-1}$ when $\mathrm{G}_t^{(N)}$ and $\boldsymbol{O}_t$ (if any) are given in stage 1.

In parallel, Liu *et al.* in [267] proposed an incremental TT method called iTTD to factorize tensors having one temporal mode. Specifically, iTTD considers coming data streams as individual tensors and then factorizes them into TT-cores. The results are appended to old estimates derived from past observations. In [268], Wang *et al.* also developed an incremental TT method called AITT to decompose tensors from industrial IoT data streams. By exploiting a relationship between the directly reshaped matrix and integration of tensor unfolding matrices, AITT can estimate effectively the underlying TT-cores. However, the two frameworks of iTTD and AITT are not really online streaming learning ones but incremental batch learning. Therefore, they are not useful for data streams from dynamical observations in time-varying environments.

5.6.2 Streaming Block-Term Decomposition

The block-term decomposition (BTD) unifies the two well-known CP and Tucker decompositions, and thus, the tracking algorithms under the CP and
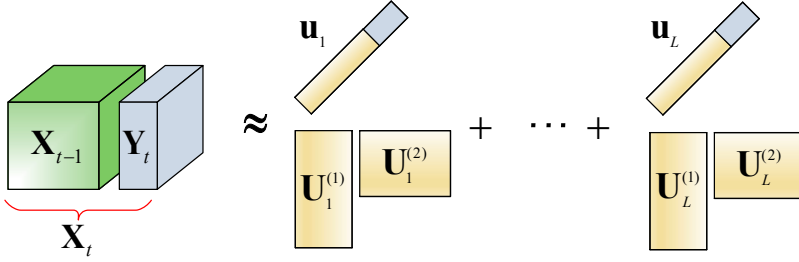
Figure 5.9: Tracking the rank-$(L, L, 1)$ BTD of 3-rd order streaming $\mathcal{X}_t$.

Tucker formats principally belong to the class of the streaming BTD with one block. When the number of blocks is greater than 2, there are only two BTD methods able to deal with streaming tensors, including OnlineBTD [269] and O-BTD-RLS [270].

The former method was proposed by Gujral *et al.* in [269] for tracking tensors under the generalized BTD format of L blocks and a multilinear rank-$(r_1, r_2, \ldots, r_N)$. On the arrival of the temporal slice $\mathcal{Y}_t$, OnlineBTD performs the following minimization:

$$\underset{\{\mathcal{G}_i\}_{i=1}^r, \{\mathbf{U}^{(n)}\}_{n=1}^N}{\operatorname{argmin}} \left\| \mathcal{Y}_t - \sum_{i=1}^r [\![\mathcal{G}_i, \{\mathbf{U}_i^{(n)}\}_{n=1}^N]\!] \right\|_F^2, \tag{5.55}$$

where $\mathbf{U}^{(n)} = \left[\mathbf{U}_1^{(n)}, \mathbf{U}_2^{(n)}, \ldots, \mathbf{U}_r^{(n)}\right]$ with $\mathbf{U}_i^{(n)} \in \mathbb{R}^{I_n \times r_n}$ and $\mathcal{G}_i \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_N}$, $1 \leq n \leq N, 1 \leq i \leq r$. Here, $\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}$ are supposed to remain unchanged with time except the last tensor factor $\mathbf{U}^{(N)}$. Prior information of L and rank-$(r_1, r_2, \ldots, r_N)$ are known in advance. Old estimates of the core tensors and tensor factors of $\mathcal{X}_{t-1}$ are used as a "warm start" for OnlineBTD at each time t. To speed up the tracking, OnlineBTD utilizes (i) an accelerated matricized tensor times Kronecker product, (ii) the pseudo-inverse operator using LU decomposition, and (iii) a dynamic programming strategy introduced by Zhou *et al.* in [175] to avoid the re-computation of duplicated Kronecker products.

The second method was introduced by Rontogiannis *et al.* in [270]. Specifically, O-BTD-RLS is designed for tracking the low rank-$(r, r, 1)$ terms of three-order streaming tensors (i.e., $r_1 = r_2 = r$ and $r_3 = 1$), see Fig. 5.9 for an illustration. In particular, the tensor factors of the underlying tensor are

incrementally updated by minimizing the following objective function:

$$\underset{\{\mathbf{U}^{(n)}\}_{n=1}^3}{\text{argmin}} \sum_{\tau=1}^t \beta^{t-\tau} \left\| \mathbf{Y}_\tau - \mathbf{U}^{(1)} \mathbf{W}_\tau \left[\mathbf{U}^{(2)} \right]^\top \right\|_F^2 + \rho_1 \sqrt{\left\| \Xi \mathbf{u}_l \right\|_2^2 + \eta^2}$$

$$+ \rho_2 \sum_{l=1}^L \sum_{k=1}^r \sqrt{\left\| \mathbf{u}_{l,k}^{(1)} \right\|_2^2 + \left\| \mathbf{u}_{l,k}^{(2)} \right\|_2^2 + \eta^2}, \tag{5.56}$$

Here, $\mathbf{U}^{(n)} = \left[\mathbf{U}_1^{(n)}, \mathbf{U}_2^{(n)}, \dots, \mathbf{U}_L^{(n)} \right]$ with $\mathbf{U}_l^{(n)} \in \mathbb{R}^{I_n \times r}$ is the n-th tensor factor of interest, and $\mathbf{u}_{l,k}^{(n)}$ is the k-th column of $\mathbf{U}_l^{(n)}$, $n = 1, 2$; $\mathbf{u}^\tau$ and $\mathbf{u}_l$ are the τ-th row and l-th column of the temporal factor $\mathbf{U}^{(3)} \in \mathbb{R}^{t \times L}$, respectively; $\mathbf{W}_\tau = \text{diag}(\mathbf{u}^\tau) \otimes \mathbf{I}_r$ and $\Xi = \text{diag}(\beta^{t-1}, \dots, \beta, 1)$; ρ_1 and ρ_2 are two regularization parameters; and η^2 is a small positive number to promote smoothness at zero. Here, the former term of (5.56) has the form of weighted least-squares while two latter terms are regularizations. Accordingly, an efficient recursive least-squares solver was introduced to minimize (5.56) effectively. An appealing feature of O-BTD-RLS is that it has the ability to reveal the BTD ranks over time by specifying the number of columns of the tensor factors which are non-negligible in magnitude at each time t.

5.6.3   Streaming t-SVD Decomposition

Similar to TT and BTD, streaming t-SVD is still in its early stage. In the literature, there exists only two works of Zhang *et al.* in [271] and Gilman *et al.* in [272, 273] addressing the problem of tensor tracking under the t-SVD format.

In [271], Zhang *et al.* introduced an online tensor PCA for sequential 2D data based on the t-SVD structure. When $\boldsymbol{\mathcal{Y}}_t$ arrives, the proposed algorithm updates:

■ The coefficient matrix $\boldsymbol{\mathcal{W}}_t$ and the sparse outlier $\boldsymbol{O}_t$ from solving the following minimization

$$\{\boldsymbol{\mathcal{W}}_t, \boldsymbol{O}_t\} = \underset{\boldsymbol{\mathcal{W}}, \boldsymbol{O}}{\text{argmin}} \frac{1}{2} \left\| \boldsymbol{\mathcal{Y}}_t - \boldsymbol{\mathcal{U}}_{t-1} * \boldsymbol{\mathcal{W}} - \boldsymbol{O} \right\|_F^2 + \frac{\lambda_1}{2} \left\| \boldsymbol{\mathcal{W}} \right\|_F^2 + \lambda_2 \left\| \boldsymbol{O} \right\|_1. \tag{5.57}$$

■ The low tubal-rank tensor $\boldsymbol{\mathcal{U}}_t$ (a.k.a. basis dictionary) from taking iFFT of the tensor $\hat{\boldsymbol{\mathcal{U}}}_t$ along the third dimension where $\hat{\boldsymbol{\mathcal{U}}}_t$ is specifically derived from

$$\hat{\mathbf{U}}_t = \underset{\hat{\mathbf{U}}}{\text{argmin}} \frac{1}{2} \text{tr} \left[\hat{\mathbf{U}}^\top (\hat{\mathbf{A}}_t + I_3 \lambda_1 \mathbf{I}) \hat{\mathbf{U}} \right] - \text{tr} \left[\hat{\mathbf{U}}^\top \hat{\mathbf{B}}_t \right]. \tag{5.58}$$

Here, $\hat{\mathbf{A}}_t = \text{diag}(\text{FFT}(\boldsymbol{\mathcal{A}}_t))$ with $\boldsymbol{\mathcal{A}}_t = \boldsymbol{\mathcal{A}}_{t-1} + \boldsymbol{W}_t * \boldsymbol{W}_t^\top$, $\hat{\mathbf{B}}_t = \text{diag}(\text{FFT}(\boldsymbol{\mathcal{B}}_t))$ with $\boldsymbol{\mathcal{B}}_t = \boldsymbol{\mathcal{B}}_{t-1} + (\boldsymbol{\mathcal{Y}}_t - \boldsymbol{O}_t) * \boldsymbol{W}_t^\top$, and the solution $\hat{\mathbf{U}}_t$ is a matricization of $\hat{\boldsymbol{\mathcal{U}}}_t$.

As the online tensor PCA above is not designed for handling missing data, Gilman *et al.* in [272, 273] proposed another algorithm called TOUCAN which is capable of tracking tensors from missing observations. Specifically, the authors proposed to solve the constrained minimization

$$\{\mathbf{U}_t, \mathbf{w}_t\} = \underset{\mathbf{U}, \mathbf{w}}{\text{argmin}} \sum_{\tau=1}^{t} \left\| \mathcal{F}_{\Omega_\tau} (\mathbf{y}_\tau - \mathbf{U} \mathbf{w}_\tau) \right\|_2^2 \quad \text{subject to } \mathbf{U}^\top \mathbf{U} = \mathbf{I}_{rI_3}, \quad (5.59)$$

where $\mathbf{y}_\tau = \texttt{unfold}(\boldsymbol{\mathcal{Y}}_\tau) \in \mathbb{C}^{I_1 I_3 \times 1}$, $\mathbf{w}_\tau = \texttt{unfold}(\boldsymbol{W}_\tau) \in \mathbb{C}^{rI_3 \times 1}$, $\mathcal{F}_{\Omega_\tau} = \mathcal{P}_{\Omega_\tau} (\mathbf{F}_{I_3}^{-1} \otimes \mathbf{I}_{I_1}) \in \mathbb{C}^{|\Omega_\tau| \times I_1 I_3}$ is the subsampled inverse Fourier transform, $\mathbf{F}_n \in \mathbb{C}^{n \times n}$ denotes the Discrete Fourier Transform matrix, the mixing matrix $\mathbf{U} \in \mathbb{R}^{I_1 I_3 \times rI_3}$ is defined as $\mathbf{U} = (\mathbf{F}_{I_3} \otimes \mathbf{I}_{I_1}) \texttt{bcirc}(\boldsymbol{\mathcal{U}}) \mathbf{F}_{I_3}^{-1}$.

Motivated by the so-called GROUSE algorithm for subspace tracking in [72], TOUCAN applies the incremental gradient descent on the tensor Grassman manifold to track $\boldsymbol{\mathcal{U}}_t$ with time. It is worth noting that the objective function (5.59) is very common in subspace tracking problems. Therefore, we can apply any subspace tracking algorithms which are capable of dealing with missing data to minimize (5.59) effectively.

5.7 Applications

Tensor tracking or dynamic tensor analysis has already been found several online applications and this section provides some typical examples in different research fields, from computer vision and neuroscience to anomaly detection.

5.7.1 Computer Vision

We begin this section with one of the earliest and most popular applications of tensor tracking: visual tracking which is an important task in computer vision [274]. Naturally, video datasets can be represented as 4-th order streaming tensors of dimensionality, width $\times$ height $\times$ channel $\times$ time. Accordingly, there are several studies devoted to developing tensor-based visual trackers for better modeling the appearance of target objects, such as [238, 275–277], to name a few. For example, Hu *et al.* in [238] proposed the so-called IRTSA tracker using incremental tensor subspace learning to capture the appearance of objects. Zhang *et al.* in [275] introduced another visual tracker called DTAMU which stands for dynamic tensor analysis with mean update. Weiming *et al.* in [276] developed a semi-supervised tensor-based visual tracker

using graph embedding. Khan *et al.* in [277] built an online spatio-temporal tensor learning model for visual tracking using Bayesian inference. It is worth noting that most of the existing tensor-based visual trackers correspond to the streaming Tucker decomposition and its variants.

Another notable application of tensor tracking in computer vision is video background and foreground separation which is quite related to visual tracking, but with a different aim of modeling the scene background and detecting the information of changes in the scene. Similar to visual tracking, many tensor-based separators were proposed, such as [27, 35, 263, 278, 279]. Particularly in [27], Thanh *et al.* proposed a robust adaptive CP method called RACP which is capable of modeling video background and detecting moving objects. Li *et al.* in [263] introduced an online robust low-rank tensor modeling (ORLTM) method and found its success in video background subtraction. Andrews *et al.* in [278] developed an online stochastic tensor decomposition for background subtraction in multispectral video sequences. A robust streaming tensor-train algorithm was developed in [35] which also has the potential to detect foreground in video. Salut *et al.* in [279] proposed an online tensor robust principal component analysis and validated its effectiveness with the problem of background and foreground separation.

In parallel, there are other interesting computer vision applications of dynamic tensor analysis, such as visual data recovery [176, 280], online video denoising [281, 282], and segmentation/classification [252, 283].

5.7.2 Neuroscience

The brain can be viewed as a complex system with various interacting regions that can produce large multivariate data over time [284]. Many types of brain data can be represented by tensors, such as electroencephalography (EEG), magnetoencephalography (MEG), functional magnetic resonance imaging (fMRI), and near-infrared spectroscopy (NIRS) [285]. Apart from three intrinsic modes (i.e., frequency, channel, and time), brain data can have higher-order modes, such as, subjects, conditions, and trials [285]. Together with the fact that brain activities can change over time, dynamic tensor analysis has become an useful tool to study the structure and function of brain from such data.

In what follows, we list some appealing brain-computer interface applications to demonstrate the use of dynamic tensor analysis in neuroscience. First, for the problem of detecting dynamic functional connectivity networks (DFCNs), Ozdemir *et al.* in [245] introduced a recursive tensor-based framework capable of tracking DFCNs over time. The proposed framework was then applied for studying error-related negativity – a brain potential response when patients make errors during cognitive tasks [286]. Mahyari *et al.* in

[287] developed a two-step approach using incremental tensor subspace analysis for detecting DFCNs. Particularly, they first detect change points at which the functional connectivity across subjects presents abrupt changes and then summarize DFCNs between successive change points. Recently, Acar *et al.* in [288] proposed to use the Parafac2 model for tracking the evolution of connectivity networks and compared its performance with ICA and IVA. For the problem of localizing dynamic brain sources over time, Ardeshir *et al.* in [289] utilized the boundary element method (BEM) [290] and the adaptive PARAFAC-RLST tracker [211] with two operational windows. A variant using augmented complex statistics in [291] also has the ability to track moving EEG sources with time. For the problem of online EEG completion, Trung *et al.* in [292] proposed an adaptive CP algorithm called NL-PETRELS capable of tracking and imputing incomplete EEG data. Thanh *et al.* in [27, 30] also demonstrated the use of ACP and RACP with real data by applying them for online EEG completion. Other neuroscience applications of tensor analysis were reviewed in [180, 293, 294].

5.7.3 Anomaly Detection

Anomaly detection, which corresponds to identifying patterns and data points that do not conform to normal behavior, plays an essential role in many applications, such as cyber security, statistics, and finance, to name a few [295]. Here, we provide some notable tensor-based anomaly detectors which are customized to specific online applications.

Shi *et al.* in [296] developed the so-called STenSr algorithm for anomaly detection and pattern discovery in spatio-temporal tensor streams from sensor networks. STenSr utilizes an incremental HOSVD and a metric based on Euclidean distance to detect abrupt changes when new data comes. Kasai *et al.* in [297] introduced an online time-structured traffic tensor tracking framework to detect network-level anomalies from link indirect measurements over time. In particular, it is based on a robust adaptive CP decomposition that uses RLS for tensor tracking and ADMM for detecting abnormal flows. Cao *et al.* in [298] designed an interactive system called Voila for detecting and monitoring visual anomalies. Voila is a tensor-based anomaly detector with an interaction design that can ranks anomalous patterns based on user input. Lin *et al.* in [299] proposed a novel method called TBAD to localize anomalous events. TBAD employs a spatial-feature-temporal tensor model and analyses latent patterns through unsupervised learning. Xu *et al.* in [300] introduced a tensor-based framework, namely SWTF, capable of detecting multiple types of anomalies in road networks. We refer the readers to [200] for a broader interdisciplinary survey of tensors for anomaly detection.

5.7.4 Others

Apart from online applications in the domains above, tensor tracking also found success in some other research fields, namely wireless communications (e.g., channel tracking [301], DOA tracking [302], and time delay estimation [303]), network analysis (e.g., link prediction [304], internet scale monitoring [305], and bot activities and network intrusions [306]), data analytics of chemical and biological manufacturing processes and components [307, 308], performance monitoring [309, 310], and transportation [311, 312].

5.8 Conclusions

Tensor tracking has recently gained increasing attention as a powerful tool for multidimensional data stream analysis. In this survey, we have provided a technical overview of online techniques for tracking streaming tensors over time. We highlighted the two most popular streaming CP and Tucker decompositions. Specifically, four main groups of streaming CP decomposition algorithms were emphasized, including subspace-based, block-coordinate descent, Bayesian inference, and multi-aspect streaming decompositions. We categorized the current streaming Tucker decomposition methods into three major classes based on their model architecture. They are online tensor dictionary learning, tensor subspace tracking, and multi-aspect streaming decompositions. Recent years have also witnessed significant advances in other types of tensor decomposition such as tensor-train, BTD, and t-SVD. A brief survey on the existing methods which are capable of tracking tensors under these formats was presented.

Robust Tensor Tracking with Missing Data and Outliers

6

Tensor decomposition is a powerful multilinear algebra tool for analyzing multiway data and has been used for various signal processing and machine learning applications. When the underlying tensor is derived from (multidimensional) data streams, streaming tensor decomposition or tensor tracking is required. In this chapter, we propose three novel adaptive algorithms for tracking the low-rank approximation of high-order streaming tensors over time, including ACP, ATD, and RACP. Under the CP format, ACP minimizes an exponentially weighted recursive least-squares cost function to obtain the tensor CP factors in an efficient way, thanks to the alternative minimization framework and the randomized sketching technique. Under the Tucker format, ATD first tracks the underlying low-dimensional subspaces covering the tensor factors, and then estimates the core tensor using a stochastic approximation. Both the two algorithms ACP and ATD are fast and fully capable of tracking streaming tensors from incomplete observations. When observations are corrupted by sparse outliers, we introduce the so-called RACP algorithm robust to gross corruptions. Particularly, RACP first performs online outlier rejection to accurately detect and remove sparse outliers, and then performs tensor factor tracking to efficiently update the tensor factors. Convergence analysis for three algorithms are established in the sense that the sequence of generated solutions converges asymptotically to a stationary point of the objective function. Extensive experiments are conducted on both synthetic and real data to demonstrate the effectiveness of the proposed algorithms in comparison with state-of-the-art adaptive algorithms.

6.1 Introduction

The era of "Big Data", which deals with massive datasets, has brought new analysis techniques for discovering new valuable information hidden in the data [313]. Among these techniques is multilinear low-rank approximation (LRA) of matrices and tensors, which has recently attracted much attention from engineers and researchers [11].

A tensor is a multidimensional array and provides a natural representation of multivariate and high-dimensional data. Low-rank approximation of tensors (t-LRA) can be considered as a multiway extension of LRA of matrices (which are two-way) to higher dimensions [10]. Generally, t-LRA is referred to as tensor decomposition which allows factorizing a tensor into a sequence of basic components [10]. As a result, t-LRA provides a useful tool for dealing with several large-scale multidimensional problems in modern data analysis which would be, otherwise, intractable by classical methods. Two widely-used approaches for t-LRA are CANDECOMP/PARAFAC (CP) decomposition[1] [14] and Tucker decomposition [314]. Under CP decomposition, a tensor can be represented as a sum of rank-1 tensors; each rank-1 tensor is formulated as the outer product of vectors. Under Tucker decomposition, a tensor is factorized into a sequence of factor matrices acting on a reduced-size core tensor. "Workhorse" algorithms are based on the method of alternating least-squares (ALS). The readers are referred to the work of [10] for a good review.

Characteristics of big data are associated with the following three "V"s: high volume, high velocity and high veracity [313]. Velocity and veracity are the focus of this chapter. Velocity requires (near) real-time processing of data streams, while veracity demands robust algorithms to better deal with missing, noisy and inconsistent data. In online applications, data acquisition is often a time-varying process in which data are serially collected or changing with time. Besides, missing data are ubiquitous and more and more common in high-dimensional problems in which collecting all attributes of data is either too expensive or even impossible. In addition, outliers which are data points that appear to be inconsistent with or exhibit abnormal behaviour different from others causes cause several issues (e.g., they introduce bias in estimation) for knowledge discovery from data in general and data streams in particular. However, well-known t-LRA algorithms either face high complexity or operate in batch mode and, thus, may not be suitable for such problems. This has led to defining a variant of t-LRA, namely tensor tracking or

---

[1]In the literature, there exist some other names for the CP decomposition: PARAFAC (Parallel Factors), CPD (Canonical Polyadic Decomposition), and CANDECOMP or CAND (Canonical Decomposition).

streaming tensor decomposition.

6.1.1 Related Works

In the literature, there are several studies related to the problem of tracking online t-LRA in the missing data context; the tensors are said to be both *streaming* and *incomplete*. For adaptive CP decomposition, Mardani *et al.* proposed TeCPSGD [106], which is a first-order algorithm and uses the method of stochastic gradient descent (SGD). Leveraging the framework of alternating minimization, TeCPSGD can estimate directly all factors but the one corresponding to the dimension growing over time in an efficient way. However, it often suffers from a slow convergence rate inherent to SGD and, hence, is not suitable for fast time-varying scenarios. To overcome this drawback, Kasai developed OLSTEC [176], which is an efficient second-order algorithm and exploits the recursive least-squares technique. OLSTEC provides a competitive performance in terms of estimation accuracy, but its computational complexity is much higher than that of TeCPSGD. In parallel, Chinh *et al.* proposed to first track the low-dimensional tensor subspace and then derive the loading factors from its Khatri-Rao structure [215]. However, the performance of this algorithm is sensitive to initialization. None of the abovementioned algorithms is capable of tracking online t-LRA when the tensors are of higher orders (i.e., greater than or equal to 4). On the other hand, some adaptive CP algorithms, such as [175,216], are capable of handling higher-order streaming tensors. However, they do not handle incomplete datasets. Recently, Zhang *et al.* have developed BRST [214], which is able to handle outliers. To track and separate the low-rank and sparsity components of the underlying tensor, a Bayesian statistical model was applied. The computational complexity of BRST is, however, very high and thus the method becomes inefficient when handling high-dimensional and fast-arriving data streams.

For adaptive Tucker decomposition, Kasai and Mishra introduced RP-Tucker [244], dealing with dynamic tensor completion. Leveraging a specific Riemannian metric, RPTucker effectively performs preconditioned SGD on the Riemannian manifold of the subspace spanned by tensor factors. Very recently, Gilman and Balzano have proposed TOUCAN (tensor rank-one update on the complex Grassmannian) [273], for tensor singular-value decomposition (t-SVD). Similar to RPTucker, TOUCAN also performs the incremental gradient descent on the Grassmann manifold. However, both algorithms are only suitable for third-order tensors. Dimitris *et al.* have recently proposed the first robust online Tucker decomposition that can deal with streaming tensors in the presence of outliers [254]. However, it was not designed for handling missing data. Some studies have been conducted to design efficient t-SVD algorithms for higher-order tensors, for example [315–317]. These al-

gorithms were designed for batch computation and thus are not suitable for dynamic models. Recently, Thanh *et al.* have proposed TT-FOA [32], which is an adaptive tensor-train (TT) model for streaming tensors. Although TT-FOA and its stochastic version are capable of tracking the online low-rank tensor-train representation of large-scale and higher-order tensors, they were not designed to handle the situation with missing data.

In the multi-aspect streaming perspective of tensor analysis, Song *et al.* proposed an effective multi-aspect streaming tensor framework (MAST) [232], used for dynamic tensor completion. MAST can successfully track the multilinear LRA of incomplete tensors with dynamic growth in more than one tensor mode. A robust version of MAST for handling outliers, called outlier-robust multi-aspect streaming tensor completion and factorization (OR-MSTC), was proposed in [233]. Thanks to the framework of alternating direction method of multipliers (ADMM), OR-MSTC can estimate the low-rank component from measurements corrupted by outliers. A new inductive framework, called SIITA, has been proposed to incorporate side information into incremental tensor analysis [248]. SIITA can be seen as a counterpart of MAST for multi-aspect streaming Tucker decomposition. Although all these approaches provide good frameworks for the problem of dynamic tensor completion, they are either useful for third-order tensors only or are of high complexity and hence relatively inefficient in applications with online data streams. In addition, convergence analysis of these algorithms is not available.

Some other studies attempted to extend robust subspace learning/online PCA for high-order tensor data. Hu *et al.* proposed an incremental tensor subspace learning algorithm, called IRTSA, and applied it to robust visual tracking in video streams [239]. Li *et al.* presented a robust algorithm that can update the tensor dictionary and detect anomalies in an online manner, namely RTSL [259]. Sobral *et al.* introduced an online stochastic tensor algorithm for learning low-rank structure and sparse components in the tensor data [278]. Another incremental tensor decomposition was designed for video background and foreground separation in [318]. Li *et al.* developed an adaptive algorithm for robust low-rank tensor learning, called ORLTM [263]. Very recently, Dimitris *et al.* have proposed the first robust online Tucker decomposition that can deal with streaming tensors in the presence of outliers [254]. However, none of the above algorithms are designed for handling missing data. The problem of robust tensor tracking for high-order incomplete streaming tensors remains largely unexplored.

6.1.2 Main Contributions

The main contributions of this chapter are summarized as follows:

- Firstly, under the CP format, we propose a novel adaptive CP (ACP) al-

gorithm for tracking higher-order incomplete streaming tensors. ACP is fast and requires a low computational complexity and memory storage, thanks to the alternative minimization and randomized sketching. It can handle incomplete tensors derived from infinite data streams because it performs CP decomposition with constant time and space complexity that are independent of time index t. A convergence analysis is then provided to establish performance guarantees. To the best of our knowledge, the proposed ACP algorithm is the first one capable of dealing with streaming tensors of higher orders with "provable" convergence guarantee.

■ Secondly, under the Tucker format, we propose the second algorithm, namely adaptive Tucker decomposition (ATD), more flexible than ACP, for the problem of online t-LRA. ATD exhibits competitive performance in terms of both estimation accuracy and computational complexity. Its convergence guarantee is also presented. Also, this chapter presents for the first time a provable adaptive Tucker algorithm for this problem.

■ Thirdly, we propose a novel method for robust adaptive CP, called RACP, for the robust tensor tracking problem in the presence of both missing data and outliers. Particularly, RACP aims to learn low-rank components of streaming tensors in an online fashion as well as offering robustness against gross data corruptions. RACP is a scalable and effective online CP algorithm with ability to (i) estimate low-rank components of streaming tensors derived from imperfect and noisy data streams due to missing observations and outlier corruptions, (ii) adapt the changes of the underlying data streams in dynamic and nonstationary environments, (iii) separate and reject sparse outliers in an online fashion with high accuracy, and (iv) easily incorporate prior information for dealing with specific constraints on the tensor model, e.g., smoothness and nonnegativity. Also, we prove that RACP is a *provable* adaptive CP algorithm with a convergence guarantee. Under mild conditions, we prove that the sequence of solutions generated by RACP converges asymptotically to a stationary point of the empirical loss function. Moreover, the asymptotic variation of the solutions and the almost-sure convergence of the objective function values are also analyzed.

■ Last but not least, we provide several experiments on both synthetic and real data to illustrate the effectiveness of the proposed algorihtms in comparison with state-of-the-art tensor tracking algorithms.

6.2   Tensor Tracking with Missing Data

6.2.1   Problem Statement

In this section, we investigate the problem of tracking an incomplete stream-
ing tensor $\boldsymbol{\mathcal{X}}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N^t}$ fixing all but the last dimension I_N^t (see illustra-
tion in Fig. 6.1 where the gray boxes represent missing data). Specifically, the
t-th tensor slice $\boldsymbol{\mathcal{Y}}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1}}$ of $\boldsymbol{\mathcal{X}}_t$ is supposed to be generated under
the following model:

$$\boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{P}}_t \circledast \left(\boldsymbol{\mathcal{L}}_t + \boldsymbol{\mathcal{N}}_t \right), \tag{6.1}$$

where $\boldsymbol{\mathcal{P}}_t$ is a binary observation mask, $\boldsymbol{\mathcal{N}}_t$ is a Gaussian noise tensor of the
same size with $\boldsymbol{\mathcal{Y}}_t$, and $\boldsymbol{\mathcal{Y}}_t$ is the multilinear low-rank component. The mask
$\boldsymbol{\mathcal{P}}_t$ shows whether the $(i_1, i_2, \ldots, i_{N-1})$-th entry of $\boldsymbol{\mathcal{Y}}_t$ is missing or not, i.e.,

$$p_{i_1 i_2 \ldots i_{N-1}} = \begin{cases} 1, & \text{if } y_{i_1 i_2 \ldots i_{N-1}} \text{ is observed,} \\ 0, & \text{otherwise.} \end{cases} \tag{6.2}$$

The low-rank component $\boldsymbol{\mathcal{Y}}_t$ is given by[2]

$$\boldsymbol{\mathcal{L}}_t \stackrel{\Delta}{=} \left[\!\left[\boldsymbol{\mathcal{G}}; \left\{ \mathbf{U}^{(n)} \right\}_{n=1}^{N-1}, \mathbf{u}_t^{(N)} \right]\!\right], \tag{6.3}$$

where $\mathbf{r} = [r_1, r_2, \ldots, r_N]$ is the desired low multilinear rank, $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_N}$
is the core tensor, $\boldsymbol{\mathcal{U}} = \{\mathbf{U}^{(n)}\}_{n=1}^{N-1}$ with $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times r_n}$ contains the first N
loading factors, and $\mathbf{u}_t^{(N)} \in \mathbb{R}^{r_N}$ is the weight vector.[3] The underlying tensor
$\boldsymbol{\mathcal{X}}_t$ is derived from appending the new slice $\boldsymbol{\mathcal{X}}_t$ to the previous $\boldsymbol{\mathcal{X}}_{t-1}$ along
the time dimension, i.e.,

$$\boldsymbol{\mathcal{X}}_t = \boldsymbol{\mathcal{X}}_{t-1} \boxplus_N \boldsymbol{\mathcal{Y}}_t, \tag{6.4}$$

where $I_N^t = I_N^{t-1} + 1$, as shown in Fig. 6.1.

The problem of tracking t-LRA of the incomplete streaming tensor $\boldsymbol{\mathcal{X}}_t$ can
be stated as follows:

> **Tensor Tracking with Missing Data**: At each time t, we observe
> a streaming tensor slice $\boldsymbol{\mathcal{Y}}_t$ under the data model (6.1). We aim to
> estimate $\boldsymbol{\mathcal{G}}_t$ and $\boldsymbol{\mathcal{U}}_t$ that will provide a good multilinear low-rank
> approximation for $\boldsymbol{\mathcal{X}}_t$ in time.

---

[2]In online setting, the tensor core $\boldsymbol{\mathcal{G}}$ and loading factors $\{\mathbf{U}^{(n)}\}$ might be changing slowly
over time, i.e., $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{G}}_t$ and $\mathbf{U}^{(n)} = \mathbf{U}_t^{(n)}, n = 1, 2, \ldots, N-1$. Our algorithms are capable of
estimating $\boldsymbol{\mathcal{G}}$ and $\boldsymbol{\mathcal{U}}$ accurately, but also successfully tracking their variation along the time.

[3]In batch setting, the weight vector $\mathbf{u}_t$ in (6.3) is seen as the t-th row of the last loading
factor $\mathbf{U}^{(N)} \in \mathbb{R}^{I_N^t \times r_N}$ of $\boldsymbol{\mathcal{X}}_t$.
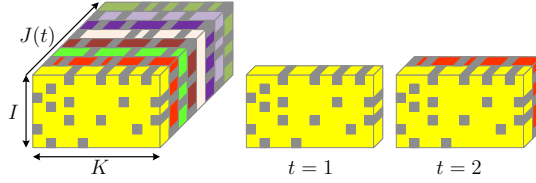
Figure 6.1: Incomplete streaming tensors.

Applying batch methods to $\boldsymbol{X}_t$ is possible, but these turns out inefficient for online (adaptive) settings. Our goal is to develop efficient one-pass algorithms, both in computational complexity and memory storage, for tracking the t-LRA of $\boldsymbol{X}_t$ from past estimations at each time t.

In an adaptive scheme, we propose to minimize the following exponentially weighted cost function:

$$\{\boldsymbol{G}_t, \boldsymbol{\mathcal{U}}_t\} = \underset{\boldsymbol{G}, \boldsymbol{\mathcal{U}}}{\operatorname{argmin}} \left[f_t(\boldsymbol{G}, \boldsymbol{\mathcal{U}}) = \frac{1}{t} \sum_{\tau=1}^{t} \beta^{t-\tau} \ell(\boldsymbol{G}, \boldsymbol{\mathcal{U}}, \boldsymbol{P}_\tau, \boldsymbol{Y}_\tau) \right], \quad (6.5)$$

where the loss function $\ell(\cdot)$ with respect to the τ-th temporal slice $\boldsymbol{Y}_\tau$ is given by

$$\ell(\boldsymbol{G}, \boldsymbol{\mathcal{U}}, \boldsymbol{P}_\tau, \boldsymbol{Y}_\tau) \overset{\Delta}{=} \min_{\mathbf{u}_\tau \in \mathbb{R}^{r_N}} \left\| \boldsymbol{P}_\tau \circledast \left(\boldsymbol{Y}_\tau - [\![\boldsymbol{G}, \{\mathbf{U}^{(n)}\}_{n=1}^{N-1}, \mathbf{u}_\tau^{(N)}]\!] \right) \right\|_F^2, \quad (6.6)$$

and $\beta \in (0, 1]$ is the forgetting parameter. Here, all observations (i.e. tensor slices) in the time interval $[1, t]$ are taken into consideration in the estimation of the underlying low-rank component at each time t. The least-squares loss $\ell(.)$ defines the residual for each observation which measures the difference between the observed value and the estimated value of the tensor slice. β is used for discounting the effect of past observations exponentially, and ensuring that observations in the distant past are substantially down-weighted in the cost function relative to the latest ones. Accordingly, when $\beta < 1$, this can facilitate the tracking ability of estimators, especially in time-varying and non-stationary environments. The effective window length for $\beta < 1$ is $(1 - \beta)^{-1}$ when t is large.

In the next two sections, we describe the two proposed algorithms for solving (6.5) under CP and Tucker decompositions. We make the following four assumptions for the convenience of deploying our algorithms as well as analyzing their performance.

(A1) Observed tensor slices $\{\boldsymbol{Y}_t\}_{t \geq 1}$ are independent and identically distributed from a data-generating distribution, which is the underlying distribution of the dataset, having a compact set $\mathcal{V}$. This assumption

is very common for convergence analysis in online settings in general and adaptive tensor decomposition in particular, e.g., [25, 106, 120, 176]. Naturally, it holds in several scenarios, for instance, real-life data are often bounded such as image, video and audio.[4]

(A2) Tensor slices $\{\boldsymbol{\mathcal{Y}}_t\}_{t \geq 1}$ follow the data model (6.47) where the true underlying loading factors $\{\mathbf{U}_t^{(n)}\}_{t \geq 1}$ are bounded, i.e., $\left\|\mathbf{U}_t^{(n)}\right\|_F \leq \kappa < \infty$. When (A1) holds, (A2) naturally holds. It also prevents arbitrarily large values in $\mathbf{U}_t^{(n)}$ and ill-conditioned computation.

(A3) Observation mask tensors $\{\boldsymbol{\mathcal{P}}_t\}_{t \geq 1}$ are independent of $\{\boldsymbol{\mathcal{Y}}_t\}_{t \geq 1}$ and their entries obey the uniform distribution. With respect to the imputation of missing values and recovery of low-rank components, the uniform randomness allows the sequence of binary masks $\{\boldsymbol{\mathcal{P}}_t\}_{t \geq 1}$ to admit stable recovery [319]. Moreover, the number of observed entries in $\boldsymbol{\mathcal{Y}}_t$ is supposed to be larger than the lower bound $O(rL \log L)$, where $L = I_1 I_2 \ldots I_{N-1}$ and $r = \max(r_1, r_2, \ldots, r_N)$, and every row of $\underline{\mathbf{Y}}_t^{(n)}$ is observed at least r entries for all n. The constraints are fundamental conditions to prevent the underdetermined imputation problem [320].

(A4) The low multilinear-rank model is either static or changing slowly over time, i.e., the core tensor and loading factors may vary slowly between two consecutive times $t - 1$ and t: $\boldsymbol{\mathcal{G}}_t \simeq \boldsymbol{\mathcal{G}}_{t-1}$ and $\mathbf{U}_t^{(n)} \simeq \mathbf{U}_{t-1}^{(n)}$. The tensor rank is supposed to be known.

6.2.2 Adaptive CP Decomposition

In this subsection, we first propose a fast adaptive CP algorithm for tracking online t-LRA of incomplete streaming tensors, called ACP. Then, we provide a performance analysis in terms of complexity and convergence to demonstrate its effectiveness and efficiency.

6.2.2.1 Proposed ACP Algorithm

Under the CP tensor model, (6.5) can be rewritten as follows:

$$\boldsymbol{\mathcal{U}}_t = \underset{\boldsymbol{\mathcal{U}}}{\operatorname{argmin}} \left[f_t(\boldsymbol{\mathcal{U}}) = \frac{1}{t} \sum_{\tau=1}^t \beta^{t-\tau} \ell(\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{P}}_\tau, \boldsymbol{\mathcal{Y}}_\tau) \right], \tag{6.7}$$

---

[4]Indeed, (A1) is a strong assumption in our analysis, but it can be relaxed as follows: Observed tensor slices $\{\boldsymbol{\mathcal{Y}}_t\}_{t \geq 1}$ are Frobenius-norm bounded, i.e., $\|\boldsymbol{\mathcal{Y}}_t\|_F < M < \infty$. Low-rank components $\{\boldsymbol{\mathcal{Y}}_t\}_{t \geq 1}$ of the observed tensor slices $\{\boldsymbol{\mathcal{Y}}_t\}_{t \geq 1}$ are supposed to be deterministic and bounded. Noise tensors $\{\boldsymbol{\mathcal{N}}_t\}_{t \geq 1}$ are i.i.d. from a distribution having a compact support.

where the loss function $\ell(\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{P}}_\tau, \boldsymbol{\mathcal{Y}}_\tau)$ is defined by

$$\ell(\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{P}}_\tau, \boldsymbol{\mathcal{Y}}_\tau) \triangleq \min_{\mathbf{u}_\tau^{(N)} \in \mathbb{R}^r} \left\| \boldsymbol{\mathcal{P}}_\tau \circledast \left(\boldsymbol{\mathcal{Y}}_\tau - \left[\!\left[\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}, \mathbf{u}_\tau^{(N)} \right]\!\right] \right) \right\|_F^2. \tag{6.8}$$

Leveraging past estimations of the loading factors, we propose to minimize the surrogate $g_t(\boldsymbol{\mathcal{U}})$ of $f_t(\boldsymbol{\mathcal{U}})$ instead, which is defined, for a given value of $\{\mathbf{u}_\tau^{(N)}\}_{1 \le \tau \le t}$, by

$$g_t(\boldsymbol{\mathcal{U}}) = \frac{1}{t} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \boldsymbol{\mathcal{P}}_\tau \circledast \left(\boldsymbol{\mathcal{Y}}_\tau - \left[\!\left[\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}, \mathbf{u}_\tau^{(N)} \right]\!\right] \right) \right\|_F^2. \tag{6.9}$$

The main motivation here stems from the following observations which will be detailed later in our convergence analysis. First, it is easy to verify that $g_t(\boldsymbol{\mathcal{U}})$ provides an upper bound on $f_t(\boldsymbol{\mathcal{U}})$ (i.e., $f_t(\boldsymbol{\mathcal{U}}) \le g_t(\boldsymbol{\mathcal{U}})$ for all $\boldsymbol{\mathcal{U}}$ and a fixed set of $\{\mathbf{u}_\tau^{(N)}\}_{1 \le \tau \le t}$). Also, the error function $e_t(\boldsymbol{\mathcal{U}}) = g_t(\boldsymbol{\mathcal{U}}) - f_t(\boldsymbol{\mathcal{U}})$ is L-smooth for some constant $L > 0$, i.e. it is differentiable and $\nabla e_t(\boldsymbol{\mathcal{U}})$ is L-Lipschitz continuous. As a result, $g_t(\boldsymbol{\mathcal{U}})$ is a *first-order surrogate* function of $f_t(\boldsymbol{\mathcal{U}})$ [321] and hence its theoretical convergence results can be achieved without making any strong assumptions on $f_t(\boldsymbol{\mathcal{U}})$. In particular, the sequence of surrogate values $\{g_t(\boldsymbol{\mathcal{U}}_t)\}_{t=1}^{\infty}$ is quasi-martingale and converges almost surely. Accordingly, under a simple assumption that the directional derivative of f_t exists in any direction at any $\boldsymbol{\mathcal{U}}$, $\{g_t(\boldsymbol{\mathcal{U}}_t)\}_{t=1}^{\infty}$ and $\{g_t(\boldsymbol{\mathcal{U}}_t)\}_{t=1}^{\infty}$ converge to the same limit. Indeed, the solution $\boldsymbol{\mathcal{U}}_t$ derived from minimizing $g_t(\boldsymbol{\mathcal{U}})$ converges to a stationary point of $f_t(\boldsymbol{\mathcal{U}})$ when t approaches infinity. Furthermore, $g_t(\boldsymbol{\mathcal{U}})$ can be effectively minimized with a convergence rate of $O(1/t)$ and it is much simpler than minimizing $f_t(\boldsymbol{\mathcal{U}})$.

In order to obtain a low-complexity estimator, we exploit that (6.9) can be efficiently solved using the alternating minimization framework whose iteration step coincides with the tensor slice's acquisition in time. In particular, it can be divided into two main stages: (i) estimate $\mathbf{u}_t^{(N)}$ first, given the old estimation $\boldsymbol{\mathcal{U}}_t$, and (ii) update the loading factor $\mathbf{U}_t^{(n)}$, given $\mathbf{u}_t^{(N)}$ and the remaining factors. The proposed ACP algorithm is summarized in Algorithm 6. In the following, we will describe the key steps of our algorithm for minimizing (6.9).

Step 1: Estimation of $\mathbf{u}_t^{(N)}$

Under the assumption that the loading factors might be static or slowly time-varying, i.e., $\boldsymbol{\mathcal{U}}_t \simeq \boldsymbol{\mathcal{U}}_{t-1}$, the weight vector $\mathbf{u}_t^{(N)}$ can be derived from the loss function $\ell(.)$ in (6.8) at time t by

$$\mathbf{u}_t^{(N)} = \operatorname*{argmin}_{\mathbf{u} \in \mathbb{R}^r} \left\| \boldsymbol{\mathcal{P}}_t \circledast \left(\boldsymbol{\mathcal{Y}}_t - \boldsymbol{\mathcal{H}}_t \times_N \mathbf{u}^\top \right) \right\|_2^2, \tag{6.10}$$

where $\mathcal{H}_t = \mathcal{I} \prod_{n=1}^{N-1} \times_n \mathbf{U}_{t-1}^{(n)}$. Problem (6.10) can be readily converted into the standard form of

$$\mathbf{u}_t^{(N)} = \operatorname*{argmin}_{\mathbf{u} \in \mathbb{R}^r} \left\| \mathbf{P}_t \left(\mathbf{y}_t - \mathbf{H}_t \mathbf{u} \right) \right\|_2^2, \tag{6.11}$$

where $\mathbf{P}_t = \texttt{diag}(\texttt{vec}(\mathcal{P}_t))$, $\mathbf{y}_t = \texttt{vec}(\mathcal{Y}_t)$, and $\mathbf{H}_t$ has the Khatri-Rao structure, i.e.,

$$\mathbf{H}_t = \bigodot_{n=1}^{N-1} \mathbf{U}_{t-1}^{(n)}. \tag{6.12}$$

For the sake of convenience, let Ω_t and $\mathbf{x}_{\Omega_t}$ be the set and vector containing the observed entries of $\mathcal{Y}_t$, while $\mathbf{H}_{\Omega_t}$ is the sub-matrix of $\mathbf{H}_t$ obtained by selecting the rows corresponding to $\mathbf{x}_{\Omega_t}$.

Generally, problem (6.11) is an overdetermined least-squares (LS) regression and requires $O(|\Omega_t| r^2)$ with respect to (w.r.t.) computational complexity to compute the exact LS solution [322]. Thus, it costs time and effort when dealing with high-dimensional and high-order tensors.

We propose to solve a regularized least-squares sketch of (6.11) instead, i.e.,

$$\mathbf{u}_t^{(N)} = \operatorname*{argmin}_{\mathbf{u} \in \mathbb{R}^r} \left\| \mathcal{L} \left(\mathbf{y}_{\Omega_t} - \mathbf{H}_{\Omega_t} \mathbf{u} \right) \right\|_2^2 + \alpha \|\mathbf{u}\|_2^2, \tag{6.13}$$

where α is a small positive parameter for regularization, $\mathcal{L}(.)$ is a sketching map that helps reduce the sample size, and hence speed up the calculations. Here, the introduction of $\alpha \|\mathbf{u}\|_2^2$ is for avoiding the singular/ill-posed computation or pathological cases as well as increasing the least-squares interpretability in practice.[5] Accordingly, the updated rule for $\mathbf{u}_t$ is given by

$$\mathbf{u}_t^{(N)} = \left(\mathbf{H}_{\mathcal{S}_t}^\top \mathbf{H}_{\mathcal{S}_t} + \alpha \mathbf{I} \right)^{-1} \mathbf{H}_{\mathcal{S}_t}^\top \mathbf{x}_{\mathcal{S}_t}, \tag{6.14}$$

where $\mathbf{H}_{\mathcal{S}_t}$ and $\mathbf{x}_{\mathcal{S}_t}$ are transformed versions of $\mathbf{H}_{\Omega_t}$ and $\mathbf{x}_{\Omega_t}$ under the sketching $\mathcal{L}(.)$, respectively.

In what follows, we indicate that in many cases, the uniform row-sampling can provide a good sketch for (6.11) in which each row has equal chance of being selected. We start by revisiting the definition of the leverage scores and coherence of a matrix.

---

[5]The value of α can be chosen in the range $[10^{-3}, 1]$ for reasonable performance in practice.

Definition 4 (Leverage Scores & Coherence [323, Definition 2.1]).
Given a matrix $\mathbf{A} = [\mathbf{a}_1^\top; \ldots ; \mathbf{a}_m^\top] \in \mathbb{R}^{m \times r}$ *with* $m > r$, *its i-th row leverage score is defined as*

$$\mathcal{T}_i(\mathbf{A}) \overset{\Delta}{=} \mathbf{a}_i^\top \left(\mathbf{A}^\top \mathbf{A} \right)^\# \mathbf{a}_i = \left\| \mathbf{U}_A(i,:) \right\|_2^2, \quad i = 1, 2, \ldots, m. \tag{6.15}$$

Here, $\mathbf{U}_A \in \mathbb{R}^{m \times r}$ *is the left singular vector matrix of* $\mathbf{A}$. *The coherence of* $\mathbf{A}$ *is the largest leverage score*

$$\mu(\mathbf{A}) = \max_i \; \mathcal{T}_i(\mathbf{A}). \tag{6.16}$$

The leverage score $\mathcal{T}_i(\mathbf{A})$ evaluates the contribution of $\mathbf{a}_i$ in constituting $\mathbf{A}$'s row space. Accordingly, if the value of $\mu(\mathbf{A})$ is high, $\mathbf{A}$ contains at least one "strong" row whose removal would have a pernicious effect on its row space. When the value of $\mu(\mathbf{A})$ is small (e.g. $\mu(\mathbf{A}) \approx r/m \ll 1$), no specific row is more important than others, i.e. information is approximately uniformized across all rows. In such a case, the matrix $\mathbf{A}$ is called incoherent. The following proposition indicates that the Khatri-Rao structure of $\mathbf{H}_t$ may increase the incoherence from its factors.

Proposition 11 (Coherence of $\mathbf{H}_t$) *Let* $\bar{\mu}_{t-1} = \dfrac{1}{N-1} \displaystyle\sum_{n=1}^{N-1} \mu\big(\mathbf{U}_{t-1}^{(n)}\big)$.

We have

$$\mu(\mathbf{H}_t) = \mu\left(\bigodot_{n=1}^{N-1} \mathbf{U}_{t-1}^{(n)} \right) \overset{(i)}{\leq} \prod_{n=1}^{N-1} \mu\big(\mathbf{U}_{t-1}^{(n)}\big) \overset{(ii)}{\leq} \bar{\mu}_{t-1}^{N-1} < 1. \tag{6.17}$$

Proof. The first inequality (i) is indeed a corollary of Lemma 4 in [324] which shows that $\mu(\mathbf{A}_1 \odot \mathbf{A}_2) \leq \mu(\mathbf{A}_1)\mu(\mathbf{A}_2)$ for any $\mathbf{A}_1$ and $\mathbf{A}_2$ of suitable sizes.

The second inequality (ii) is obtained by applying the AM–GM inequality to the set of N positive numbers $\big\{\mu\big(\mathbf{U}_{t-1}^{(n)}\big)\big\}_{n=1}^{M-1}$.

Accordingly, when dealing with a high-order streaming tensor (N is large) and/or with some incoherent tensor factors, $\mu(\mathbf{H}_t) \leq \bar{\mu}_{t-1}^{N-1} \ll \bar{\mu}_{t-1} < 1$, i.e., $\mathbf{H}_t$ has low coherence. In such cases, uniform row-sampling is effective [325, 326]. In the presence of highly coherent factors, a preconditioning (mixing) step is necessary to guarantee the incoherence. For instance, the subsampled randomized Hadamard transform (SRHT) is a good candidate which can produce a transformed matrix whose rows have (almost) uniform leverage scores [327]. In this context, we here emphasize that well-known randomized LS algorithms can help save much computational complexity while obtaining

reasonable estimations of $\mathbf{u}_t^{(N)}$, especially for large-scale low-rank tensors.

Step 2: Estimation of $\mathrm{U}_t^{(n)}$

The loading factor $\mathbf{U}_t^{(n)}$ can be updated by minimizing the objective function $g_t(\cdot)$ w.r.t. $\mathbf{U}^{(n)}$, as

$$\mathbf{U}_t^{(n)} = \underset{\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times r}}{\mathrm{argmin}} \ \frac{1}{t} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \underline{\mathbf{P}}_\tau^{(n)} \circledast \left(\underline{\mathbf{Y}}_\tau^{(n)} - \mathbf{U}^{(n)} \big(\mathbf{W}_\tau^{(n)} \big)^\top \right) \right\|_F^2, \qquad (6.18)$$

where $\underline{\mathbf{Y}}_\tau^{(n)}$ (resp. $\underline{\mathbf{P}}_\tau^{(n)}$) is the mode-$n$ unfolding of $\boldsymbol{\mathcal{Y}}_\tau$ (resp. $\boldsymbol{\mathcal{P}}_\tau$) and the coefficient matrix $\mathbf{W}_\tau^{(n)}$ is given by

$$\mathbf{W}_\tau^{(n)} = \left(\overset{N-1}{\underset{i=1, i \neq n}{\bigodot}} \mathbf{U}_{t-1}^{(i)} \right) \odot (\mathbf{u}_\tau^{(N)})^\top. \qquad (6.19)$$

Interestingly, we exploit the fact that minimization (6.18) can boil down to the problem of subspace tracking in the presence of missing data [41]. Particularly, the solution of (6.18) can be obtained by minimizing subproblems for each row $\mathbf{u}_m^{(n)}$ of $\mathbf{U}^{(n)}$, $m = 1, 2, \ldots, I_n$ as

$$\mathbf{u}_{t,m}^{(n)} = \underset{\mathbf{u}_m^{(n)} \in \mathbb{R}^r}{\mathrm{argmin}} \ \frac{1}{t} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \underline{\mathbf{P}}_{\tau,m}^{(n)} \left(\big(\underline{\mathbf{y}}_{\tau,m}^{(n)} \big)^\top - \mathbf{W}_\tau^{(n)} \big(\mathbf{u}_m^{(n)} \big)^\top \right) \right\|_F^2, \qquad (6.20)$$

where $\underline{\mathbf{y}}_{\tau,m}^{(n)}$ is the m-th row of $\underline{\mathbf{Y}}_\tau^{(n)}$ and the row-mask matrix $\underline{\mathbf{P}}_{\tau,m}^{(n)} = \mathrm{diag}(\underline{\mathbf{P}}_\tau^{(n)}(m,:))$. Thanks to the parallel scheme of the well-known PETRELS algorithm for subspace tracking [73], we derive an efficient estimator for minimizing the exponentially weighted LS cost function (6.18). Particularly, we first define two auxiliary matrices $\mathbf{S}_t^{(n)}$ and $\mathbf{V}_t^{(n)}$ as follows[6]

$$\mathbf{S}_t^{(n)} = \beta \mathbf{S}_{t-1}^{(n)} + \big(\mathbf{W}_t^{(n)} \big)^\top \mathbf{W}_t^{(n)}, \qquad (6.21)$$

$$\mathbf{V}_t^{(n)} = \big(\mathbf{S}_t^{(n)} \big)^{-1} \big(\mathbf{W}_t^{(n)} \big)^\top. \qquad (6.22)$$

The loading factor $\mathbf{U}_t^{(n)}$ is then updated recursively by

$$\mathbf{U}_t^{(n)} = \mathbf{U}_{t-1}^{(n)} + \Delta \underline{\mathbf{Y}}_t^{(n)} \big(\mathbf{V}_t^{(n)} \big)^\top, \qquad (6.23)$$

where the matrix $\Delta \underline{\mathbf{Y}}_t^{(n)}$ is derived from the mode-n unfolding of the residual error tensor $\Delta \boldsymbol{\mathcal{Y}}_t$

$$\Delta \boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{P}}_t \circledast \big(\boldsymbol{\mathcal{Y}}_t - \boldsymbol{\mathcal{H}}_t \times_N (\mathbf{u}_t^{(N)})^\top \big). \qquad (6.24)$$

---

[6]To enable the recursive updating rule, the matrix $\mathbf{S}_0^{(n)}$ is initialized by a scaled identity matrix $\mathbf{S}_0^{(n)} = \delta_n \mathbf{I}_{r_n}$ with $\delta_n > 0$.

INPUT: Incomplete slices $\left\{\mathcal{P}_t \circledast \boldsymbol{\mathcal{Y}}_t\right\}_{t=1}^{\infty}$, $\boldsymbol{\mathcal{Y}}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times 1}$, CP rank r, Forgetting factor $\beta \in (0, 1]$, Parameters: $\alpha > 0$, $\delta > 0$, and $m > 0$.
INITIALIZATION: $\left\{\mathbf{U}_0^{(n)}\right\}_{n=1}^{N-1}$ is initialized randomly and $\left\{\mathbf{S}_0^{(n)}\right\}_{n=1}^{N-1} = \delta \mathbf{I}_r$.
MAIN PROGRAM:

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

    $\boldsymbol{\mathcal{Y}}_{\Omega_t} = \mathcal{P}_t \circledast \boldsymbol{\mathcal{Y}}_t$

    Step 1: Estimation of $\mathbf{u}_t$

$$\mathcal{S} = \texttt{randsample}\left(|\Omega_t|, \lfloor mr \log r \rfloor\right)$$

$$\mathbf{H}_t = \mathcal{I} \prod_{n=1}^{N-1} \times_n \mathbf{U}_{t-1}^{(n)}$$

$$\mathbf{u}_t = \left(\mathbf{H}_{\mathcal{S}_t}^\top \mathbf{H}_{\mathcal{S}_t} + \alpha \mathbf{I}\right)^{-1} \mathbf{H}_{\mathcal{S}_t}^\top \mathbf{y}_{\mathcal{S}_t}$$

$$\mathbf{U}_t^{(N)} = \left[\mathbf{U}_{t-1}^{(N)^\top}, \mathbf{u}_t^{(N)}\right]^\top$$

$$\Delta \boldsymbol{\mathcal{Y}}_t = \mathcal{P}_t \circledast \left(\boldsymbol{\mathcal{Y}}_t - \mathcal{H}_t \times_N \mathbf{u}_t^\top\right)$$

    Step 2: Estimation of $\left\{\mathbf{U}_t^{(n)}\right\}_{n=1}^{N-1}$

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

$$\underline{\mathbf{Y}}_{\Omega_t}^{(n)} = \texttt{unfold}_n(\boldsymbol{\mathcal{Y}}_{\Omega_t})$$

$$\Delta \underline{\mathbf{Y}}_t^{(n)} = \texttt{unfold}_n(\Delta \boldsymbol{\mathcal{X}}_t)$$

$$\mathbf{W}_t^{(n)} = \left(\left(\mathbf{U}_{t-1}^{(n)}\right)^{\#} \underline{\mathbf{Y}}_{\Omega_t}^{(n)}\right)^\top$$

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

$$\mathbf{V}_t^{(n)} = \left(\mathbf{S}_t^{(n)}\right)^{-1} \mathbf{W}_t^{(n)}$$

$$\mathbf{U}_t^{(n)} = \mathbf{U}_{t-1}^{(n)} + \Delta \underline{\mathbf{Y}}_t^{(n)} \left(\mathbf{V}_t^{(n)}\right)^\top$$

    end

end
OUTPUT: $\left\{\mathbf{U}_t^{(n)}\right\}_{n=1}^{N}$

Algorithm 6: Adaptive CP Decomposition (ACP)

This is not PETRELS, but a modified version. Here, we can utilize the already updated $\mathbf{U}_t^{(n)}$ for tracking the remaining factors which can improve the rate of convergence. Also, we can estimate all the N factors in a parallel scheme which reduces further the cost when several computational units are available.

6.2.2.2 Performance Analysis

Memory Storage and Computational Complexity: For the sake of simplifying the analysis, we assume that the fixed dimensions of the streaming tensor $\boldsymbol{\mathcal{Y}}_t$ are equal to I and the CP rank is much lower than I, $r \ll I$.

With respect to memory storage, ACP requires $O\big((N-1)(Ir+r^2)\big)$ words of memory at each time t, in particular for N loading factors $\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}$ and $N-1$ matrices $\mathbf{S}_t^{(n)}$ of size $r \times r$.

In terms of computational complexity, the estimation of $\mathbf{u}_t^{(N)}$ costs $O(|\mathcal{S}_t|r^2)$ flops from solving the randomized LS regression and forming the sketch for $\mathbf{H}_{\mathcal{S}_t}$. The complexity for updating the loading factor $\mathbf{U}_t^{(n)}$ comes from the computation of the two matrices $\Delta\underline{\mathbf{Y}}_t^{(n)}$ and $\mathbf{V}_t^{(n)}$. In particular, the first one requires $O(|\Omega_t|r)$ flops while the latter costs $O(I^{N-2}r^2)$ flops. Note that, the matrix $\mathbf{S}_t^{(n)}$ is of size $r \times r$, thus the computation of $(\mathbf{S}_t^{(n)})^{-1}$ is not expensive and it is independent of the tensor dimension. In conclusion, the overall computational complexity is $O\big(|\Omega_t|r + ((N-1)I^{N-2} + |\mathcal{S}_t|)r^2\big)$ flops and reduces to $O\big(|\Omega_t|r + (I^{N-2} + |\mathcal{S}_t|)r^2\big)$ flops in a parallel scheme. Note that when a preconditioning step (e.g. SRHT) is needed to guarantee the incoherence of $\mathbf{H}_{\Omega_t}$, ACP requires an additional cost of $O\big(|\Omega_t|r\log r\big)$ flops [328].

Convergence Guarantee: Inspired by our companion work on robust subspace tracking in [25] and the convergence analysis for 3-order tensors in [106, 176], we derive a unified approach to analyze the convergence behavior of ACP for high-order streaming tensors with missing data. Specifically, we analyze the convergence of both the sequence of objective values $\{f_t(\boldsymbol{\mathcal{U}}_t)\}_{t=1}^{\infty}$ and the sequence of generated solutions $\{\boldsymbol{\mathcal{U}}_t\}_{t=1}^{\infty}$. Our main theoretical result is stated in the following lemma.

> **Lemma 9** *Given assumptions (A1)-(A4), $\beta = 1$, and the true $\boldsymbol{\mathcal{U}}$ is fixed, the sequence of solutions $\{\mathbf{U}_t\}_{t=1}^{\infty}$ generated by ACP converges to a minimum point of f_t when $t \to \infty$.*

Proof Sketch. Our proof contains three main stages: (S1) we show that the solutions $\{\boldsymbol{\mathcal{U}}_t, \mathbf{u}_t\}_{t=1}^{\infty}$ are uniformly bounded to justify the well-definedness condition. Their variations between two successive time instances satisfy $\|\mathbf{U}_{t+1}^{(n)} - \mathbf{U}_t^{(n)}\|_F \to O(1/t)$ a.s. (S2) The sequence of nonnegative surrogate values $\{g_t(\boldsymbol{\mathcal{U}}_t)\}_{t=1}^{\infty}$ is quasi-martingale and convergent almost surely. (S3) The empirical loss function $\{f_t(\boldsymbol{\mathcal{U}}_t)\}_{t=1}^{\infty}$ and its surrogate $\{g_t(\boldsymbol{\mathcal{U}}_t)\}_{t=1}^{\infty}$ converge to the same limit, i.e., $g_t(\boldsymbol{\mathcal{U}}_t) \to f_t(\boldsymbol{\mathcal{U}}_t)$ a.s. Accordingly, $\{\boldsymbol{\mathcal{U}}_t\}_{t=1}^{\infty}$ converges to a stationary point of $f_t(\boldsymbol{\mathcal{U}})$, i.e., $\nabla f_t(\boldsymbol{\mathcal{U}}_t) \xrightarrow{t\to\infty} 0$. Details of the analysis is provided in the Appendix A.

INPUT: Observations $\{\mathcal{P}_t \circledast \mathcal{X}_t\}_{t=1}^{\infty}$, $\mathcal{X}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times 1}$, Tucker rank $\mathbf{r}_{\mathrm{TD}} = [r_1, \ldots, r_N]$, Forgetting factor β, Parameters: $\alpha > 0$, $\delta > 0$, and $m > 0$.

INITIALIZATION: $\{\mathbf{U}_0^{(n)}\}_{n=1}^{N-1}$ and $\mathcal{G}_0$ are initialized randomly, $\{\mathbf{S}_0^{(n)}\}_{n=1}^{N-1} = \delta \mathbf{I}_{r_n}$.

MAIN PROGRAM:

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

$\quad \mathcal{Y}_{\Omega_t} = \mathcal{P}_t \circledast \mathcal{Y}_t$

$\quad$ **Step 1: Estimation of** $\mathbf{u}_t$

$$\mathcal{S} = \mathtt{randsample}\big(|\Omega_t|, \lfloor mr_N \log r_N \rfloor\big)$$

$$\mathcal{H}_t = \mathcal{G}_{t-1} \prod_{n=1}^{N-1} \times_n \mathbf{U}_{t-1}^{(n)}$$

$$\mathbf{u}_t = \big(\mathbf{H}_{\mathcal{S}_t}^\top \mathbf{H}_{\mathcal{S}_t} + \alpha \mathbf{I}\big)^{-1} \mathbf{H}_{\mathcal{S}_t}^\top \mathbf{y}_{\mathcal{S}_t}$$

$$\mathbf{U}_t^{(N)} = \Big[\mathbf{U}_{t-1}^{(N)\top}, \mathbf{u}_t^{(N)}\Big]^\top$$

$$\Delta \mathcal{Y}_t = \mathcal{P}_t \circledast \big(\mathcal{Y}_t - \mathcal{H}_t \times_N \mathbf{u}_t^\top\big)$$

$\quad$ **Step 2: Estimation of** $\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}$

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

$$\mathbf{W}_t^{(n)} = \big(\mathbf{U}_{t-1}^{(n)}\big)^{\#} \underline{\mathbf{Y}}_{\Omega_t}^{(n)}$$

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

$$\mathbf{V}_t^{(n)} = \big(\mathbf{S}_t^{(n)}\big)^{-1} \mathbf{W}_t^{(n)}$$

$$\mathbf{U}_t^{(n)} = \mathbf{U}_{t-1}^{(n)} + \Delta \underline{\mathbf{Y}}_t^{(n)}\big(\mathbf{V}_t^{(n)}\big)^\top$$

$\quad\quad$ **end**

$\quad$ **Step 3: Estimation of** $\mathcal{G}_t$

$$\mathbf{Z}_t = \mathbf{u}_t \otimes \left(\bigotimes_{n=2}^{N-2} \mathbf{U}_t^{(n)}\right)$$

$$\Delta \mathbf{G}_t = \big(\mathbf{U}_t^{(1)}\big)^{\#} \Delta \underline{\mathbf{Y}}_t^{(1)} \mathbf{Z}_t^{\#}$$

$$\Delta \mathcal{G}_t = \mathtt{reshape}(\Delta \mathbf{G}_t, \mathbf{r}_{\mathrm{TD}})$$

$$\mathcal{G}_t = \mathcal{G}_{t-1} + \Delta \mathcal{G}_t$$

end

OUTPUT: $\{\mathbf{U}_t^{(n)}\}_{n=1}^{N}$ and $\mathcal{G}_t$

Algorithm 7: Adaptive Tucker Decomposition (ATD)

6.2.3 Adaptive Tucker Decomposition

The proposed ACP algorithm is not always well-defined due to the fact that for a given CP rank, the optimal CP-based representation of tensors may be nonexistent [207]. Under the Tucker format, we now propose a more flexible algorithm called adaptive Tucker decomposition (ATD).

In the same way, we propose to minimize the following surrogate function

$g_t(\mathcal{G}, \mathcal{U})$ of $f_t(\mathcal{G}, \mathcal{U})$ in (6.5):

$$\{\mathcal{G}_t, \mathcal{U}_t\} = \operatorname*{argmin}_{\mathcal{G}, \mathcal{U}} g_t(\mathcal{G}, \mathcal{U}), \quad \text{where}$$

$$g_t(\mathcal{G}, \mathcal{U}) = \frac{1}{t} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \mathcal{P}_\tau \circledast \left(\mathcal{Y}_\tau - \left[\!\left[\mathcal{G}; \{\mathbf{U}^{(n)}\}_{n=1}^{N-1}, \mathbf{u}_\tau^{(N)} \right]\!\right] \right) \right\|_F^2, \quad (6.25)$$

to leverage old estimations of the tensor core and the loading factors at each time t.

6.2.3.1 Proposed ATD Algorithm

Thanks to the alternating minimization framework, we can obtain an efficient first-order estimator for optimizing (6.25) in the same manner as ACP. Specifically, we first update the weight vector $\mathbf{u}_t^{(N)}$, given old estimations of $\mathcal{G}$ and $\mathcal{U}$, then estimate the loading factors $\{\mathbf{U}_t^{(n)}\}_{n\geq 1}$ given $\mathbf{u}_t^{(N)}$, $\mathcal{G}_{t-1}$ and the remaining factors, and finally obtain the core tensor $\mathcal{G}_t$ from the latest updated factors. The proposed algorithm is summarized in Algorithm 7.

Step 1: Estimation of $\mathbf{u}_t^{(N)}$

We can derive the weight vector $\mathbf{u}_t^{(N)}$ from the minimizing the last summand of $g_t(\mathcal{G}, \mathcal{U})$ as follows:

$$\mathbf{u}_t^{(N)} = \operatorname*{argmin}_{\mathbf{u} \in \mathbb{R}^{r_N}} \left\| \mathcal{P}_t \circledast \left(\mathcal{Y}_t - \mathcal{H}_t \times_N \mathbf{u}^\top \right) \right\|_2^2, \quad (6.26)$$

where $\mathcal{H}_t = \mathcal{G}_{t-1} \prod_{n=1}^{N-1} \times_n \mathbf{U}_{t-1}^{(n)}$. Similar to (6.10), the expression (6.26) can be readily reformulated into its matrix-vector format as follows:

$$\mathbf{u}_t^{(N)} = \operatorname*{argmin}_{\mathbf{u} \in \mathbb{R}^{r_N}} \left\| \mathbf{P}_t \left(\mathbf{y}_t - \mathbf{H}_t \mathbf{u} \right) \right\|_2^2, \quad (6.27)$$

where $\mathbf{y}_t = \mathrm{vec}(\mathcal{Y}_t)$, $\mathbf{H}_t$ is the unfolding matrix of the tensor $\mathcal{H}_t$ and the observation matrix $\mathbf{P}_t = \mathrm{diag}(\mathrm{vec}(\mathcal{P}_t))$. The closed-form solution of (6.27) can be directly obtained by applying the LS method as

$$\mathbf{u}_t^{(N)} = \left(\mathbf{H}_t^\top \mathbf{P}_t \mathbf{H}_t + \alpha \mathbf{I} \right)^{-1} \mathbf{H}_t^\top \mathbf{P}_t \mathbf{y}_t, \quad (6.28)$$

where $\alpha > 0$ is a small regularization parameter to avoid pathological cases in practice.

In order to speed up the computation of (6.28), the same randomized sampling technique as in (7.12) can be applied to obtain an approximated version of $\mathbf{u}_t^{(N)}$.

Step 2: Estimation of $\mathbf{U}_t^{(n)}$

Given $\mathbf{u}_t$ and old estimations of $\boldsymbol{\mathcal{G}}_{t-1}$ and $\boldsymbol{\mathcal{U}}_{t-1}$, we rewrite the minimization (6.25) with respect to the variable $\mathbf{U}^{(n)}$ as follows:

$$\mathbf{U}_t^{(n)} = \underset{\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times r_n}}{\operatorname{argmin}} \frac{1}{t} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \underline{\mathbf{P}}_\tau^{(n)} \circledast \left(\underline{\mathbf{Y}}_\tau^{(n)} - \mathbf{U}^{(n)} \mathbf{W}_\tau^{(n)} \right) \right\|_F^2, \qquad (6.29)$$

where the coefficient matrix $\mathbf{W}_\tau^{(n)}$ is the mode-n unfolding of the tensor $\boldsymbol{\mathcal{W}}_\tau$ which is defined by

$$\boldsymbol{\mathcal{W}}_\tau = \left(\boldsymbol{\mathcal{G}}_{t-1} \prod_{i=1, i \neq n}^{N-1} \times_i \mathbf{U}_{t-1}^{(i)} \right) \times_{N+1} (\mathbf{u}_\tau^{(N)})^\top. \qquad (6.30)$$

Minimization (6.29) is similar to its counterpart in the proposed ACP algorithm in (6.18). Therefore, we can apply the same subspace-based technique to update $\mathbf{U}_t^{(n)}$. In particular, the updating rule for $\mathbf{U}_t^{(n)}$ can be given by

$$\mathbf{U}_t^{(n)} = \mathbf{U}_{t-1}^{(n)} + \Delta \underline{\mathbf{Y}}_t^{(n)} (\mathbf{V}_t^{(n)})^\top, \qquad (6.31)$$

where the residual error $\Delta \underline{\mathbf{Y}}_t^{(n)}$ and the coefficient matrix $\mathbf{V}_t^{(n)}$ are computed as

$$\Delta \underline{\mathbf{Y}}_t^{(n)} = \underline{\mathbf{P}}_t^{(n)} \circledast \left(\underline{\mathbf{Y}}_t^{(n)} - \mathbf{U}_{t-1}^{(n)} \mathbf{W}_t^{(n)} \right), \qquad (6.32)$$

$$\mathbf{V}_t^{(n)} = \left(\mathbf{S}_t^{(n)} \right)^{-1} \mathbf{W}_t^{(n)}, \qquad (6.33)$$

where the matrix $\mathbf{S}_t^{(n)}$ is updated recursively as

$$\mathbf{S}_t^{(n)} = \beta \mathbf{S}_{t-1}^{(n)} + \mathbf{W}_t^{(n)} (\mathbf{W}_t^{(n)})^\top. \qquad (6.34)$$

Step 3: Estimation of $\boldsymbol{\mathcal{G}}_t$

For the estimation of $\boldsymbol{\mathcal{G}}_t$ given the latest updated loading factors, (6.25) is reformulated as

$$\boldsymbol{\mathcal{G}}_t = \underset{\boldsymbol{\mathcal{G}}}{\operatorname{argmin}} \frac{1}{t} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \underline{\mathbf{P}}_\tau^{(1)} \circledast \left(\underline{\mathbf{Y}}_\tau^{(1)} - \mathbf{U}_t^{(1)} \underline{\mathbf{G}}^{(1)} \mathbf{Z}_\tau \right) \right\|_F^2, \qquad (6.35)$$

where the variable $\underline{\mathbf{G}}^{(1)}$ is the mode-1 unfolding of $\boldsymbol{\mathcal{G}}$ and the matrix $\mathbf{Z}_\tau$ is given by

$$\mathbf{Z}_\tau = \mathbf{u}_\tau^{(N)} \otimes \left(\bigotimes_{n=2}^{N-1} \mathbf{U}_t^{(n)} \right). \qquad (6.36)$$

When handling a streaming tensor with a huge number of slices (i.e., t is large) and a large number of unknown parameters in $\boldsymbol{G}$ (i.e., $\prod_{n=1}^{N} r_n$ is large), applying batch gradient methods for (6.35) may be time-consuming despite the effect of the forgetting factor λ. Stochastic approximation is introduced as a good alternative [329].

In particular, we minimize the last summand of (6.35) instead:

$$\boldsymbol{G}_t = \underset{\boldsymbol{G}}{\operatorname{argmin}} \left\| \underline{\mathbf{P}}_t^{(1)} \circledast \left(\underline{\mathbf{Y}}_t^{(1)} - \mathbf{U}_t^{(1)} \underline{\mathbf{G}}^{(1)} \mathbf{Z}_t \right) \right\|_F^2. \tag{6.37}$$

Given the estimation of $\boldsymbol{\mathcal{U}}_t$, the residual error between the newcoming tensor slice and the recovered one is given by

$$\Delta \underline{\mathbf{Y}}_t^{(1)} = \underline{\mathbf{P}}_t^{(1)} \circledast \left(\underline{\mathbf{Y}}_t^{(1)} - \mathbf{U}_t^{(1)} \underline{\mathbf{G}}_{t-1}^{(1)} \mathbf{Z}_t \right). \tag{6.38}$$

Accordingly, we can derive the variation of $\boldsymbol{G}$ at time t from

$$\Delta \underline{\mathbf{Y}}_t^{(1)} = \underline{\mathbf{P}}_t^{(1)} \circledast \left(\mathbf{U}_t^{(1)} \Delta \underline{\mathbf{G}}_t^{(1)} \mathbf{Z}_t \right), \tag{6.39}$$

where $\Delta \underline{\mathbf{G}}_t^{(1)} = \underline{\mathbf{G}}_t^{(1)} - \underline{\mathbf{G}}_{t-1}^{(1)}$. In particular, $\Delta \boldsymbol{G}_t$ is computed as[7]

$$\Delta \underline{\mathbf{G}}_t^{(1)} = \left(\mathbf{U}_t^{(1)} \right)^{\#} \Delta \underline{\mathbf{Y}}_t^{(1)} \mathbf{Z}_t^{\#}. \tag{6.41}$$

After that, $\Delta \underline{\mathbf{G}}_t^{(1)}$ will be reshaped into a tensor $\Delta \boldsymbol{G}_t$ of size $r_1 \times r_2 \times \cdots \times r_N$. To sum up, we obtain the simple rule for updating $\boldsymbol{G}_t$ as follows:

$$\boldsymbol{G}_t = \boldsymbol{G}_{t-1} + \Delta \boldsymbol{G}_t. \tag{6.42}$$

We note that for overdetermined cases, the rule for updating $\boldsymbol{G}_t$ can be sped up by using the following "vector trick" in [331]:

$$\operatorname{vec}(\mathbf{ABC}^{\top}) = (\mathbf{C} \otimes \mathbf{A}) \operatorname{vec}(\mathbf{B}). \tag{6.43}$$

In particular, the expression (6.39) can be cast into the standard least-squares format as follows:

$$\boldsymbol{\delta} \mathbf{y}_t = \mathbf{P}_t \left(\mathbf{u}_t \otimes \left(\bigotimes_{n=1}^{N-1} \mathbf{U}_t^{(n)} \right) \right) \boldsymbol{\delta} \mathbf{g}_t, \tag{6.44}$$

where $\boldsymbol{\delta} \mathbf{x}_t = \operatorname{vec}(\Delta \underline{\mathbf{Y}}_t^{(1)})$, $\boldsymbol{\delta} \mathbf{g}_t = \operatorname{vec}(\Delta \underline{\mathbf{G}}_t^{(1)})$ and $\mathbf{P}_t = \operatorname{diag}(\operatorname{vec}(\underline{\mathbf{P}}_t^{(1)}))$. Interestingly, (6.44) is of the Kronecker structure, thus $\boldsymbol{\delta} \mathbf{g}_t$ can be efficiently computed by applying randomized sketching techniques with a much lower complexity, e.g., the uniform sampling or the Kronecker product regression in [332].

---

[7]Since $\mathbf{Z}_t$ is of the Kronecker structure, we can obtain the pseudoinverse of $\mathbf{Z}_t$ efficiently by using the following nice property [330]

$$(\mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \cdots \otimes \mathbf{A}_n)^{\#} = \mathbf{A}_1^{\#} \otimes \mathbf{A}_2^{\#} \otimes \cdots \otimes \mathbf{A}_n^{\#}. \tag{6.40}$$

Step 4: Orthogonalization Step (Optional)

In the cases where the orthogonality constraints are imposed on the loading factors, we add an orthogonalization step of $\mathbf{U}^{(n)}$ at each time t as follows:

$$\mathbf{U}_t^{(n)} = \mathbf{U}_t^{(n)}\left[\left(\mathbf{U}_t^{(n)}\right)^\top \mathbf{U}_t^{(n)}\right]^{-1/2}, \tag{6.45}$$

where $(.)^{-1/2}$ represents the inverse square root or simply take the QR decomposition of $\mathbf{U}_t$. Accordingly, the update of $\Delta\underline{\mathbf{G}}_t$ in (6.41) can be speeded up by replacing the pseudo-inverse with the transpose operator:

$$\Delta\underline{\mathbf{G}}_t = \left(\mathbf{U}_t^{(1)}\right)^\top \Delta\underline{\mathbf{Y}}_t^{(1)}\mathbf{Z}_t^\top. \tag{6.46}$$

6.2.3.2 Performance Analysis

Memory Storage and Computational Complexity: We assume that the fixed dimensions of the streaming tensor are equal to I and the desired Tucker rank is $\mathbf{r}_{\text{TD}} = [r, r, \ldots, r]$.

In terms of memory storage, ATD requires $O(r^N)$ and $O((N-1)Ir)$ words of memory for saving the core tensor $\boldsymbol{\mathcal{G}}$ and $N-1$ loading factors $\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}$ respectively. In addition, the cost for saving $N-1$ matrices $\mathbf{S}_t^{(n)}$ is $O((N-1)r^2)$ words of memory in total.

In terms of computational complexity, the computation of ATD comes from three main estimations: (i) the weight vector $\mathbf{u}_t^{(N)}$, (ii) the loading factors $\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}$ and (iii) the core tensor $\boldsymbol{\mathcal{G}}$. The two former estimations are similar to that of ACP, so they require a cost of $O\big(|\Omega_t|r + (I^{N-2} + |\mathcal{S}_1|)r^2\big)$ flops in a parallel scheme where $|\mathcal{S}_1|$ denotes the size of the sampling set of (6.27). The latter estimation costs $O\big(|\Omega_t|r + I^{N-2}r^{2N}\big)$ flops for computing $\Delta\boldsymbol{\mathcal{X}}$ and $\Delta\boldsymbol{\mathcal{G}}$. If using the randomize technique in this stage, the complexity is reduced to $O\big(|\Omega_t|r + |\mathcal{S}_2|r^{2N}\big)$ flops where $\mathcal{S}_2$ is the set of selected samples from (6.44). Therefore, the overall computational complexity of ATD is $O\big(|\Omega_t|r + (I^{N-2} + |\mathcal{S}_1|)r^2 + |\mathcal{S}_2|r^{2N}\big)$ in parallel scheme.

Convergence Guarantee: The convergence of ATD can be stated by the following lemma:

> **Lemma 10** *Given assumptions (A1)-(A4), $\beta = 1$, the true $\boldsymbol{\mathcal{G}}$ and $\boldsymbol{\mathcal{U}}$ are fixed, the solutions $\{\boldsymbol{\mathcal{G}}_t, \boldsymbol{\mathcal{U}}_t\}_{t=1}^\infty$ generated by ATD converges to a stationary point of the empirical cost function f_t when $t \to \infty$.*

Proof Sketch. The proof of Lemma 10 can be obtained by applying the same arguments and principles as in the case of ACP, detailed in the Appendix A. In particular, the analysis consists of the following three main
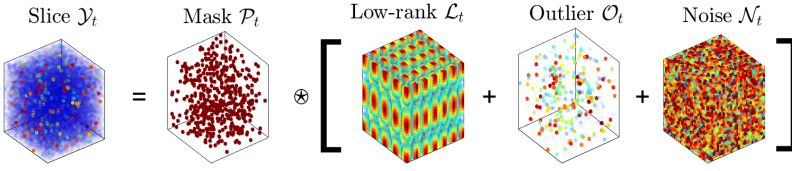
Figure 6.2: Temporal slice $\mathcal{Y}_t$ with missing data and sparse outliers.

stages: (S1) the surrogate function $g_t(\mathcal{G}, \mathcal{U})$ is strongly bi-convex in the sense that $\mathcal{G}$ and $\mathcal{U}$ are seen as multivariate variables. Solutions $\{\mathcal{G}_t, \mathcal{U}_t\}_{t=1}^{\infty}$ generated by ATD are bounded and their variations between two successive time instances satisfy $\|\mathbf{U}_{t+1}^{(n)} - \mathbf{U}_t^{(n)}\|_F \to O(1/t)$ a.s. (S2) The nonnegative sequence $\{g_t(\mathcal{G}_t, \mathcal{U}_t)\}_{t=1}^{\infty}$ is quasi-martingale and hence convergent almost surely. Furthermore, $g_t(\mathcal{G}_t, \mathcal{U}_t) - f_t(\mathcal{G}_t, \mathcal{U}_t) \to 0$ a.s. (S3) The empirical cost function $f_t(\mathcal{G}, \mathcal{U})$ is continuously differentiable and Lipschitz. The sequence of solutions $\{\mathcal{G}_t, \mathcal{U}_t\}_{t=1}^{\infty}$ converges to a stationary point of $f_t(\mathcal{G}, \mathcal{U})$, i.e., when $t \to \infty$, the gradient $\nabla f_t(\mathcal{G}_t, \mathcal{U}_t) \to 0$ a.s.

6.3   Tensor Tracking with Sparse Outliers

6.3.1   Problem Statement

Here, we consider an incomplete streaming tensor $\mathcal{X}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times t}$ whose slices are serially observed with time. At each time t, $\mathcal{X}_t$ is particularly obtained by concatenating a new incoming "slice" $\mathcal{Y}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times 1}$ into the previous $\mathcal{X}_{t-1}$ along the time dimension, i.e., $\mathcal{X}_t = \mathcal{X}_{t-1} \boxplus_N \mathcal{Y}_t$. Particularly, we suppose to observe the slice $\mathcal{Y}_t$ satisfying the following model:

$$\mathcal{Y}_t = \mathcal{P}_t \circledast (\mathcal{L}_t + \mathcal{O}_t + \mathcal{N}_t), \tag{6.47}$$

where $\mathcal{P}_t$ is a binary mask tensor, $\mathcal{L}_t$ is a low-rank tensor, $\mathcal{O}_t$ is a sparse tensor containing outliers, $\mathcal{N}_t$ is a Gaussian noise tensor, and all these tensors are of the same size with $\mathcal{Y}_t$, please see Fig 6.2 for an illustration.

Specifically, the observation mask $\mathcal{P}_t$ indicates whether the $(i_1, i_2, \ldots, i_{N-1})$-th entry of $\mathcal{L}_t$ is observed or missing, i.e.,

$$p_{i_1 i_2 \ldots i_N} = \begin{cases} 0, & \text{if } x_{i_1 i_2 \ldots i_{N-1}} \text{ is missing,} \\ 1, & \text{otherwise.} \end{cases} \tag{6.48}$$

The low-rank tensor $\mathcal{L}_t$ is generated according to the following model:

$$\mathcal{L}_t = \left[\!\left[\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}, \mathbf{u}_t^{(N)} \right]\!\right], \tag{6.49}$$

where $\mathbf{u}_t^{(N)} \in \mathbb{R}^{r \times 1}$ is a weight vector[8] and $\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}$, with $\mathbf{U}^{(n)} \in \mathcal{U}_n \subseteq \mathbb{R}^{I_n \times r}$, are loading factors. For short, write $\mathcal{D} := \mathcal{U}_1 \times \mathcal{U}_2 \times \cdots \times \mathcal{U}_N$ and denote $\mathbf{D} = \left[(\mathbf{U}^{(1)})^\top, \ldots, (\mathbf{U}^{(N)})^\top \right]^\top$ the tensor dictionary containing all loading factors. The robust tensor tracking (RTT) problem can be stated as follows:

> **Tensor Tracking with Sparse Outliers**: At each time t, we observe a streaming tensor slice $\mathcal{Y}_t$ under the data model (6.47). We aim to estimate $\mathbf{D}_t \in \mathcal{D}$ such that it can provide a good multilinear low-rank approximation for $\mathcal{X}_t$ in time.

Now, we define a loss function $\ell(\cdot)$ that not only promotes sparsity but also preserves convexity. For a fixed $\mathbf{D}$ and a tensor slice $\mathcal{X}$ under a binary observation mask $\mathcal{P}$, the loss function w.r.t. $\mathbf{D}$ and $\{\mathcal{P}, \mathcal{Y}\}$ is defined as

$$\ell(\mathbf{D}, \mathcal{P}, \mathcal{X}) = \min_{\mathbf{u}, \mathbf{O}} \tilde{\ell}(\mathbf{D}, \mathcal{P}, \mathcal{X}, \mathbf{O}, \mathbf{u}), \text{ with} \tag{6.50}$$

$$\tilde{\ell}(\mathbf{D}, \mathcal{P}, \mathcal{Y}, \mathbf{O}, \mathbf{u}) = \|\mathbf{O}\|_1 + \frac{\rho}{2} \left\| \mathcal{P} \circledast \left(\mathcal{Y} - \mathbf{O} - \mathcal{H} \times_N \mathbf{u}^\top \right) \right\|_F^2, \tag{6.51}$$

where $\mathcal{H} = \mathcal{I} \prod_{n=1}^{N-1} \times_n \mathbf{U}^{(n)}$. The ℓ_1-norm is to promote the sparsity on $\mathbf{O}$ and $\rho > 0$ is a regularized parameter.

Now, given a streaming set of incomplete tensor slices $\{\mathcal{P}_\tau \circledast \mathcal{Y}_\tau\}_{\tau=1}^t$, robust tensor tracking (RTT) can be formulated as the following optimization problem:

$$\mathbf{D}_t = \operatorname*{argmin}_{\mathbf{D}} \left[f_t(\mathbf{D}) = \frac{1}{L_t} \sum_{\tau=t-L_t+1}^{t} \beta^{t-\tau} \ell(\mathbf{D}, \mathcal{P}_\tau, \mathcal{Y}_\tau) \right], \tag{6.52}$$

where L_t is the length of a sliding window and β is a forgetting factor. When $L_t = t, \beta = 1$, the minimization of (6.52) boils down to its counterpart in batch setting. When $0 < L_t < t$ or $\beta < 1$, it reduces the impact of past observations, and hence facilitates the tracking ability of RTT estimators in time-varying conditions.

We make some assumptions to support the proposed algorithm in Section III. First, entries of tensor slices $\{\mathcal{Y}_t\}_{t \geq 1}$ are Frobenius-norm bounded, i.e., $\|\mathcal{Y}_t\|_F \leq M_x < \infty \ \forall t$. This prevents arbitrarily large values in observations and ill-conditioned computation. Next, the tensor rank r is supposed to remain unchanged over time. In addition, tensor factors $\{\mathbf{U}_t^{(n)}\}_{n=1}^N$ are bounded and full column rank, i.e., $\operatorname{rank}(\mathbf{U}_t^{(n)}) = r < I_n$ and $\|\mathbf{U}_t^{(n)}\|_F \leq \kappa_U < \infty \ \forall n$. Besides, the variation between two consecutive time instants is small, $\mathbf{U}_t^{(n)} \simeq$

---

[8]In batch setting, the weight vector $\mathbf{u}_t$ in (6.49) is seen as the t-th row of the last loading factor $\mathbf{U}^{(N)} \in \mathbb{R}^{I_N^t \times r}$ of the underlying tensor $\mathcal{X}_t$.

$\mathbf{U}_{t-1}^{(n)}$ $\forall n, t$ i.e. $\mathbf{D}_{t-1} \simeq \mathbf{D}_t$. This assumption permits the estimation of the outliers and the coefficient vector from the previous estimation with reasonable accuracy. Under these assumptions, our optimization algorithm is capable of accurately estimating tensor factors, but also successfully tracking their variation along the time.

6.3.2   Robust Adaptive CP Decomposition

In this section, we first propose the robust adaptive CP (RACP) algorithm for the RTT problem in the presence of missing data and outliers. Then, we introduce two simple extensions of RACP in order to deal with smoothness condition and nonnegative constraints.

6.3.2.1   Proposed RACP Algorithm

Solving the minimization of (6.52) exactly is possible but difficult since $f_t(\cdot)$ is nonconvex. We here adapt it using the majorization-minimization (MM) framework [321], which has been successfully applied to several signal processing problems in general [333] and online learning problems in particular [25, 120, 121, 334]. In essence, we decompose it into two main stages: (i) online outlier rejection and (ii) tensor factor tracking.

On the arrival of $\boldsymbol{\mathcal{Y}}_t$ at each time t, we first estimate the outlier tensor $\boldsymbol{\mathcal{O}}_t$ and the coefficient vector $\mathbf{u}_t$ based on the old estimation $\mathbf{D}_{t-1}$. Specifically, we solve the following optimization:

$$\{\boldsymbol{\mathcal{O}}_t, \mathbf{u}_t\} = \underset{O, \mathbf{u}}{\operatorname{argmin}} \; \tilde{\ell}\big(\mathbf{D}_{t-1}, \boldsymbol{\mathcal{P}}_t, \boldsymbol{\mathcal{Y}}_t, O, \mathbf{u}\big). \tag{6.53}$$

From the past statistics $\{\mathbf{D}_\tau, \boldsymbol{\mathcal{P}}_\tau, \boldsymbol{\mathcal{Y}}_\tau, \boldsymbol{\mathcal{O}}_\tau, \mathbf{u}_\tau\}_{\tau \geq 1}$, the set of loading factors $\mathbf{D}_t = \{\mathbf{U}_t^{(n)}\}_{n=1}^N$ can be updated by minimizing the following majorizing surrogate $\tilde{f}_t(\cdot)$:

$$\tilde{f}_t(\mathbf{D}) = \frac{1}{L_t} \sum_{\tau=t-L_t+1}^{t} \beta^{t-\tau} \tilde{\ell}\big(\mathbf{D}, \boldsymbol{\mathcal{P}}_\tau, \boldsymbol{\mathcal{Y}}_\tau, \boldsymbol{\mathcal{O}}_\tau, \mathbf{u}_\tau\big), \tag{6.54}$$

that locally approximates $f_t(\cdot)$. Note that $\tilde{f}_t(\mathbf{D})$ is not only first-order surrogate, but also a majorant function of $f_t(\mathbf{D})$, that is, for all t and $\mathbf{D}$, we always have $f_t(\mathbf{D}) \leq \tilde{f}_t(\mathbf{D})$ and the error function $e_t(\mathbf{D}) = \tilde{f}_t(\mathbf{D}) - f_t(\mathbf{D})$ is Lipschitz continuous. In fact, $\tilde{f}_t(\mathbf{D})$ and $f_t(\mathbf{D})$ converge almost-surely to the same limit, and the solution $\mathbf{D}_t$, which minimizes $\tilde{f}_t(\mathbf{D})$, is exactly the one of $f_t(\mathbf{D})$ when $t \to \infty$. The results will be later proven in our convergence analysis. In what follows, we propose two solvers for minimizing (6.53) and (6.54) efficiently.

INPUT: Tensor slices $\left\{ \mathcal{P}_t \circledast \mathcal{Y}_t \right\}_{t=1}^{\infty}$, $\mathcal{Y}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times 1}$, rank r, forgetting factor $\beta \in (0, 1]$,
Parameters: penalty $\rho > 0$, precision ϵ^{res}, $\epsilon^{out} > 0$, maximum iteration K, $\alpha \in [1.5, 1.8]$, $\delta > 0$.
INITIALIZATION: $\left\{ \mathbf{U}_0^{(n)} \right\}_{n=1}^{N-1}$ is initialized randomly and $\left\{ \mathbf{S}_0^{(n)} \right\}_{n=1}^{N-1} = \delta \mathbf{I}_r$.
MAIN PROGRAM:
for $t = 1, 2, \ldots$ **do**

 Stage 1: Online Outlier Rejection

  $\mathbf{H}_{t-1} = \bigodot_{n=1}^{N-1} \mathbf{U}_{t-1}^{(n)}$
  $\mathbf{o}^0, \mathbf{z}^0, \mathbf{u}^0 \leftarrow \mathbf{0}$

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

   $\mathbf{u}^i = \left(\mathbf{H}_{t-1}^\top \mathbf{P}_t \mathbf{H}_{t-1} \right)^{\#} \mathbf{H}_{t-1}^\top \mathbf{P}_t \left(\mathbf{y}_t - \mathbf{o}^{i-1} - \mathbf{z}^{i-1}/\rho \right)$,
   $\mathbf{r}^i = \alpha \mathbf{P}_t \left(\mathbf{y}_t - \mathbf{H}_{t-1} \mathbf{u}^i \right) + (1-\alpha) \mathbf{o}^{i-1}$
   $\mathbf{o}^i = \mathcal{S}_{1/\rho} \left(\mathbf{r}^i - \mathbf{z}^{i-1}/\rho \right)$,
   $\mathbf{z}^i = \mathbf{z}^{i-1} + \rho \left(\mathbf{o}^i - \mathbf{r}^i \right)$,
   if stopping criteria are met **break**

  end

  Outlier Removal (Re-update of $\mathcal{P}_t$ in (6.61) is optional): $\widehat{\mathcal{Y}}_t = \mathcal{P}_t \circledast \left(\mathcal{Y}_t - \mathcal{O}_t \right)$

 Stage 2: Estimation of $\left\{ \mathbf{U}_t^{(n)} \right\}_{n=1}^{N}$

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

   $\mathbf{W}_t^{(n)} = \left(\bigodot_{i=1, i \neq n}^{N} \mathbf{U}_{t-1}^{(i)} \right) \odot \left(\mathbf{u}_t^{(N)} \right)^\top$ [Jacobi]

   $\mathbf{W}_t^{(n)} = \left(\bigodot_{i=1}^{n-1} \mathbf{U}_t^{(i)} \right) \odot \left(\bigodot_{i=n+1}^{N} \mathbf{U}_{t-1}^{(i)} \right) \odot \left(\mathbf{u}_t^{(N)} \right)^\top$ [Gauss-Seidel]

   $\widetilde{\mathbf{W}}_t^{(n)} = \left[\left(\mathbf{W}_t^{(n)} \right)^\top \quad \left(\mathbf{W}_{t-L_t}^{(n)} \right)^\top \right]^\top$

   for $m = 1, 2, \ldots, I_n$ **do**

    $\underline{\widetilde{\mathbf{P}}}_{t,m}^{(n)} = \begin{bmatrix} \underline{\mathbf{P}}_{t,m}^{(n)} & \mathbf{0} \\ \mathbf{0} & -\beta^{L_t} \underline{\mathbf{P}}_{t-L_t,m}^{(n)} \end{bmatrix}$

    $\underline{\widetilde{\mathbf{Y}}}_{t,m}^{(n)} = \left[\underline{\widehat{\mathbf{Y}}}_{t,m}^{(n)} \quad \underline{\widehat{\mathbf{Y}}}_{t-L_t,m}^{(n)} \right]$

    $\mathbf{S}_{t,m}^{(n)} = \beta \mathbf{S}_{t-1,m}^{(n)} + \left(\widetilde{\mathbf{W}}_t^{(n)} \right)^\top \underline{\widetilde{\mathbf{P}}}_{t,m}^{(n)} \widetilde{\mathbf{W}}_t^{(n)}$

    $\mathbf{V}_{t,m}^{(n)} = \left(\mathbf{S}_{t,m}^{(n)} \right)^{-1} \left(\widetilde{\mathbf{W}}_t^{(n)} \right)^\top$

    $\delta \underline{\widetilde{\mathbf{y}}}_{t,m}^{(n)} = \underline{\widetilde{\mathbf{P}}}_{t,m}^{(n)} \left(\left(\underline{\widetilde{\mathbf{y}}}_{t,m}^{(n)} \right)^\top - \widetilde{\mathbf{W}}_t^{(n)} \left(\mathbf{u}_{t-1,m}^{(n)} \right)^\top \right)$

    $\mathbf{u}_{t,m}^{(n)} = \mathbf{u}_{t-1,m}^{(n)} + \left(\delta \underline{\widetilde{\mathbf{y}}}_{t,m}^{(n)} \right)^\top \left(\mathbf{V}_{t,m}^{(n)} \right)^\top$

   end

  end

 Stage 3: (Optional) Normalization and Re-estimation of $\mathbf{u}_t$

  Column-wise Normalization: $\left[\mathbf{U}_t^{(n)} \right]_{:,r} = \left[\mathbf{U}_t^{(n)} \right]_{:,r} / \left\| \left[\mathbf{U}_t^{(n)} \right]_{:,r} \right\|_2^2$.

  Re-estimation of $\mathbf{u}_t$: $\mathbf{u}_t = \left(\mathbf{H}_t^\top \mathbf{P}_t \mathbf{H}_t \right)^{\#} \mathbf{H}_t^\top \mathbf{P}_t \left(\mathbf{x}_t - \mathbf{o}_t \right)$ where $\mathbf{H}_t = \bigodot_{n=1}^{N} \mathbf{U}_t^{(n)}$

end
OUTPUT: Loading factors $\left\{ \mathbf{U}_t^{(n)} \right\}_{n=1}^{N}$.

Algorithm 8: Robust Adaptive CP Decomposition (RACP)

Stage 1: Online Outlier Rejection

To estimate O_t and $\mathbf{u}_t^{(N)}$, we recast (6.53) into the following standard matrix-vector form:

$$\{\mathbf{o}_t, \mathbf{u}_t^{(N)}\} = \operatorname*{argmin}_{\mathbf{o}, \mathbf{u}} \; \|\mathbf{o}\|_1 + \frac{\rho}{2}\left\|\mathbf{P}_t\left(\mathbf{y}_t - \mathbf{o} - \mathbf{H}_{t-1}\mathbf{u}\right)\right\|_2^2, \qquad (6.55)$$

where $\mathbf{o}_t = \operatorname{vec}(O_t)$, $\mathbf{y}_t = \operatorname{vec}(\mathcal{Y}_t)$, the observation mask matrix $\mathbf{P}_t = \operatorname{diag}(\operatorname{vec}(\mathcal{P}_t))$, and $\mathbf{H}_{t-1}$ is of a Khatri-Rao structure, i.e., $\mathbf{H}_{t-1} = \bigodot_{n=1}^{N-1} \mathbf{U}_{t-1}^{(n)}$.

Since both terms of (6.55) are convex, it can be efficiently solved by several methods with convergence guarantees. Here, we use an ADMM solver to minimize (6.55) due to its simple interpretation and moderate convergence rate [114]. At the i-th iteration, we particularly read

$$\mathbf{u}^i = \left(\mathbf{H}_{t-1}^\top \mathbf{P}_t \mathbf{H}_{t-1}\right)^{\#} \mathbf{H}_{t-1}^\top \mathbf{P}_t \left(\mathbf{y}_t - \mathbf{o}^{i-1} - \mathbf{z}^{i-1}/\rho\right), \qquad (6.56)$$

$$\mathbf{r}^i = \alpha \mathbf{P}_t\left(\mathbf{y}_t - \mathbf{H}_{t-1}\mathbf{u}^i\right) + (1-\alpha)\mathbf{o}^{i-1} \qquad (6.57)$$

$$\mathbf{o}^i = \mathcal{S}_{1/\rho}\left(\mathbf{r}^i - \mathbf{z}^{i-1}/\rho\right), \qquad (6.58)$$

$$\mathbf{z}^i = \mathbf{z}^{i-1} + \rho(\mathbf{o}^i - \mathbf{r}^i), \qquad (6.59)$$

where $\mathcal{S}(\cdot)$ is the soft-thresholding operator of the ℓ_1-norm defined as $\mathcal{S}_\alpha(x) = \max(0, x-\alpha) - \max(0, -x-\alpha)$ and $\alpha \in [1.5, 1.8]$ is a relaxation parameter. The procedure is stopped when residuals are small, i.e., $\|\mathbf{P}_t(\mathbf{y}_t - \mathbf{H}_{t-1}\mathbf{u}^i - \mathbf{o}^i)\|_2 \leq \epsilon^{res}$ and $\|\mathbf{o}^i - \mathbf{r}^i\|_2 \leq \epsilon^{out}$ where $\epsilon^{res}, \epsilon^{out} > 0$ are predefined accuracy parameters or when the procedure reaches the maximum number of iterations.

After the sparse outlier O_t is detected, we reduce the effect of O_t on the tracking process by the following outlier removal

$$\widehat{\mathcal{Y}}_t = \mathcal{P}_t \circledast (\mathcal{Y}_t - O_t). \qquad (6.60)$$

In some cases, we can skip the corrupted entries in $\mathcal{Y}_t$ by re-updating the mask $\mathcal{P}_t$ as

$$p_{i_1 i_2 \dots i_N} = \begin{cases} 0, & \text{if } x_{i_1 \dots i_N} \text{ is missing or outlier,} \\ 1, & \text{otherwise.} \end{cases} \qquad (6.61)$$

Here, the removal step (6.60) still holds under the new binary mask $\mathcal{P}_t$. This approach stems from the following observations. In the context of subspace tracking (ST), rejecting outliers can facilitate the tracking ability of ST estimators since only "clean" measurements involve the process [25]. Our next stage for estimating the tensor basis can indeed boil down to the ST problem with missing data, so the outlier rejection mechanism of (6.61) can improve performance. Please see Fig. 6.24 for an illustration that the outlier rejection mechanism can help improve the convergence rate of RACP when the fraction of corrupted entries is not too large.

Stage 2: Estimation of factors $\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}$

The optimization (6.54) can be effectively solved by using the block-coordinate descent (BCD) technique. The main idea is to minimize alternately the surrogate $\tilde{f}_t(\cdot)$ w.r.t. each factor $\mathbf{U}_t^{(n)}$ while fixing the remaining factors (hereafter denoted as $\tilde{f}_t(\mathbf{U}_t^{(n)}, .)$ for short), that is,

$$\mathbf{U}_t^{(n)} = \underset{\mathbf{U}^{(n)}}{\operatorname{argmin}} \tilde{f}_t(\mathbf{U}^{(n)}, .). \tag{6.62}$$

Minimization (6.62) is equivalent to

$$\mathbf{U}_t^{(n)} = \underset{\mathbf{U}^{(n)}}{\operatorname{argmin}} \sum_{\tau=t-L_t+1}^{t} \beta^{t-\tau} \left\| \underline{\mathbf{P}}_\tau^{(n)} \circledast \left(\widehat{\underline{\mathbf{Y}}}_\tau^{(n)} - \mathbf{U}^{(n)} (\mathbf{W}_\tau^{(n)})^\top \right) \right\|_F^2, \tag{6.63}$$

where $\widehat{\underline{\mathbf{Y}}}_\tau^{(n)}$ and $\underline{\mathbf{P}}_\tau^{(n)}$ are the mode-n unfoldings of $\widehat{\boldsymbol{\mathcal{Y}}}_\tau$ and $\boldsymbol{\mathcal{P}}_\tau$, and $\mathbf{W}_\tau^{(n)}$ is given by

$$\mathbf{W}_\tau^{(n)} = \begin{cases} \left(\overset{N-1}{\underset{i=1, i \neq n}{\bigodot}} \mathbf{U}_{t-1}^{(i)} \right) \odot (\mathbf{u}_\tau^{(N)})^\top & [\text{Jacobi}], \\ \left(\overset{n-1}{\underset{i=1}{\bigodot}} \mathbf{U}_t^{(i)} \right) \odot \left(\overset{N-1}{\underset{i=n+1}{\bigodot}} \mathbf{U}_{t-1}^{(i)} \right) \odot (\mathbf{u}_\tau^{(N)})^\top & [\text{Gauss-Seidel}]. \end{cases} \tag{6.64}$$

Depending on the implementation, we can use one of the two iterative methods: the Jacobi scheme supports the parallel and/or distributed processing while the Gauss-Seidel scheme is useful for a sequential (serial) one. Excepting the closed-form of $\mathbf{W}_\tau^{(n)}$, both methods share the same procedure for solving (6.63) which is detailed as follows.

The minimization of (6.63) can be decomposed into sub-problems for each row $\mathbf{u}_m^{(n)}$ of $\mathbf{U}^{(n)}$, $m = 1, 2, \ldots, I_n$, as

$$\mathbf{u}_{t,m}^{(n)} = \underset{\mathbf{u}_m^{(n)}}{\operatorname{argmin}} \sum_{\tau=t-L_t+1}^{t} \beta^{t-\tau} \left\| \underline{\mathbf{P}}_{\tau,m}^{(n)} \left((\hat{\mathbf{y}}_{\tau,m}^{(n)})^\top - \mathbf{W}_\tau^{(n)} (\mathbf{u}_m^{(n)})^\top \right) \right\|_F^2, \tag{6.65}$$

where $\hat{\mathbf{y}}_{\tau,m}^{(n)}$ is the m-th row of $\widehat{\underline{\mathbf{Y}}}_\tau^{(n)}$, and the row-mask matrix is given by $\underline{\mathbf{P}}_{\tau,m}^{(n)} = \operatorname{diag}\left(\underline{\mathbf{P}}_\tau^{(n)}(m, :) \right)$.

The optimal solution of (6.65) can be derived from setting its derivative to zero

$$\sum_{\tau=t-L_t+1}^{t} \beta^{t-\tau} (\mathbf{W}_\tau^{(n)})^\top \underline{\mathbf{P}}_{\tau,m}^{(n)} (\hat{\mathbf{y}}_{\tau,m}^{(n)})^\top = \sum_{\tau=t-L_t+1}^{t} \beta^{t-\tau} (\mathbf{W}_\tau^{(n)})^\top \underline{\mathbf{P}}_{\tau,m}^{(n)} \mathbf{W}_\tau^{(n)} (\mathbf{u}_{t,m}^{(n)})^\top.$$

$$\tag{6.66}$$

Instead of solving (6.66) directly, we propose a more elegant recursive way to obtain $\mathbf{u}_{t,m}^{(n)}$ as follows. First, let us denote the left hand side of (6.66) by $\mathbf{d}_{t,m}^{(n)}$, and

$$\mathbf{S}_{t,m}^{(n)} = \sum_{\tau=t-L_t+1}^{t} \beta^{t-\tau} \big(\mathbf{W}_\tau^{(n)}\big)^\top \underline{\mathbf{P}}_{\tau,m}^{(n)} \mathbf{W}_\tau^{(n)}. \tag{6.67}$$

Accordingly, (6.66) becomes

$$\mathbf{S}_{t,m}^{(n)} \big(\mathbf{u}_{t,m}^{(n)}\big)^\top = \mathbf{d}_{t,m}^{(n)}. \tag{6.68}$$

Interestingly, both $\mathbf{d}_{t,m}^{(n)}$ and $\mathbf{S}_{t,m}^{(n)}$ can be updated recursively:

$$\mathbf{d}_{t,m}^{(n)} = \beta \mathbf{d}_{t-1,m}^{(n)} + \big(\widetilde{\mathbf{W}}_t^{(n)}\big)^\top \underline{\widetilde{\mathbf{P}}}_{t,m}^{(n)} \big(\underline{\tilde{\mathbf{y}}}_{t,m}^{(n)}\big)^\top, \tag{6.69}$$

$$\mathbf{S}_{t,m}^{(n)} = \beta \mathbf{S}_{t-1,m}^{(n)} + \big(\widetilde{\mathbf{W}}_t^{(n)}\big)^\top \underline{\widetilde{\mathbf{P}}}_{t,m}^{(n)} \widetilde{\mathbf{W}}_t^{(n)}. \tag{6.70}$$

where

$$\widetilde{\mathbf{W}}_t^{(n)} = \big[\big(\mathbf{W}_t^{(n)}\big)^\top \ \big(\mathbf{W}_{t-L_t}^{(n)}\big)^\top\big]^\top, \tag{6.71}$$

$$\underline{\tilde{\mathbf{y}}}_{t,m}^{(n)} = \big[\underline{\hat{\mathbf{y}}}_{t,m}^{(n)} \ \underline{\hat{\mathbf{y}}}_{t-L_t,m}^{(n)}\big], \tag{6.72}$$

$$\underline{\widetilde{\mathbf{P}}}_{t,m}^{(n)} = \begin{bmatrix} \underline{\mathbf{P}}_{t,m}^{(n)} & \mathbf{0} \\ \mathbf{0} & -\beta^{L_t} \underline{\mathbf{P}}_{t-L_t,m}^{(n)} \end{bmatrix}. \tag{6.73}$$

Therefore, we can rewrite (6.68) as

$$\mathbf{S}_{t,m}^{(n)} \big(\mathbf{u}_{t,m}^{(n)}\big)^\top = \beta \mathbf{d}_{t-1,m}^{(n)} + \big(\widetilde{\mathbf{W}}_t^{(n)}\big)^\top \underline{\widetilde{\mathbf{P}}}_{t,m}^{(n)} \big(\underline{\tilde{\mathbf{y}}}_{t,m}^{(n)}\big)^\top$$

$$= \beta \mathbf{S}_{t-1,m}^{(n)} \big(\mathbf{u}_{t-1,m}^{(n)}\big)^\top + \big(\widetilde{\mathbf{W}}_t^{(n)}\big)^\top \underline{\widetilde{\mathbf{P}}}_{t,m}^{(n)} \big(\underline{\tilde{\mathbf{y}}}_{t,m}^{(n)}\big)^\top$$

$$= \mathbf{S}_{t,m}^{(n)} \big(\mathbf{u}_{t-1,m}^{(n)}\big)^\top + \big(\widetilde{\mathbf{W}}_t^{(n)}\big)^\top \underline{\widetilde{\mathbf{P}}}_{t,m}^{(n)} \big(\big(\underline{\tilde{\mathbf{y}}}_{t,m}^{(n)}\big)^\top - \widetilde{\mathbf{W}}_t^{(n)} \big(\mathbf{u}_{t-1,m}^{(n)}\big)^\top\big). \tag{6.74}$$

Multiplying both sides by $\big(\mathbf{S}_{t,m}^{(n)}\big)^{-1}$ results in

$$\mathbf{u}_{t,m}^{(n)} = \mathbf{u}_{t-1,m}^{(n)} + \big(\delta \underline{\tilde{\mathbf{y}}}_{t,m}^{(n)}\big)^\top \big(\mathbf{V}_{t,m}^{(n)}\big)^\top, \tag{6.75}$$

where

$$\delta \underline{\tilde{\mathbf{y}}}_{t,m}^{(n)} = \underline{\widetilde{\mathbf{P}}}_{t,m}^{(n)} \big(\big(\underline{\tilde{\mathbf{y}}}_{t,m}^{(n)}\big)^\top - \widetilde{\mathbf{W}}_t^{(n)} \big(\mathbf{u}_{t-1,m}^{(n)}\big)^\top\big), \tag{6.76a}$$

$$\mathbf{V}_{t,m}^{(n)} = \big(\mathbf{S}_{t,m}^{(n)}\big)^{-1} \big(\widetilde{\mathbf{W}}_t^{(n)}\big)^\top. \tag{6.76b}$$

Collecting all rows $\mathbf{u}_{t,m}^{(n)}$ together, $m = 1, 2, \ldots, I_n$, a simplified version of (6.75) for updating the whole factor $\mathbf{U}_t^{(n)}$ can be given by[9]

$$\mathbf{U}_t^{(n)} = \mathbf{U}_{t-1}^{(n)} + \Delta\widetilde{\mathbf{Y}}_t^{(n)}(\mathbf{V}_t^{(n)})^\top, \tag{6.77}$$

where

$$\mathbf{S}_t^{(n)} = \beta\mathbf{S}_{t-1}^{(n)} + (\widetilde{\mathbf{W}}_t^{(n)})^\top\widetilde{\mathbf{W}}_t^{(n)}, \tag{6.78a}$$

$$\mathbf{V}_t^{(n)} = (\mathbf{S}_t^{(n)})^{-1}(\widetilde{\mathbf{W}}_t^{(n)})^\top, \tag{6.78b}$$

$$\Delta\widetilde{\mathbf{Y}}_t^{(n)} = \underline{\widetilde{\mathbf{P}}}_t^{(n)} \circledast \left(\widetilde{\mathbf{Y}}_t^{(n)} - \mathbf{U}_{t-1}^{(n)}(\widetilde{\mathbf{W}}_t^{(n)})^\top\right), \tag{6.78c}$$

with $\widetilde{\mathbf{Y}}_t^{(n)} = \begin{bmatrix} \widehat{\mathbf{Y}}_t^{(n)} & \widehat{\mathbf{Y}}_{t-L_t,m}^{(n)} \end{bmatrix}$. In this way, we can skip several operations and save a memory storage of $O\left(\sum_{n=1}^{N-1}(I_n - 1)(I_n r + r^2)\right)$. Specifically, the cost of computing (6.78a) is $O\left(r^2 \prod_{i=1,i\neq n}^{N-1} I_i\right)$. The computation of (6.78b) also requires a cost of $O\left(r^2 \prod_{i=1,i\neq n}^{N-1} I_i\right)$ because $\mathbf{S}_t^{(n)}$ is of size $r \times r$ and its inverse computation is not expensive and independent of the tensor dimension. The error matrix $\Delta\widetilde{\mathbf{Y}}_t^{(n)}$ in (6.78c) can be derived from Step 1 by reshaping the residual vector $\mathbf{P}_t(\mathbf{y}_t - \mathbf{o}_t - \mathbf{H}_{t-1}\mathbf{u}_t)$. The most expensive step is the product $\Delta\widetilde{\mathbf{Y}}_t^{(n)}(\mathbf{V}_t^{(n)})^\top$ which costs $r \prod_{i=1}^{N-1} I_i$ flops while the addition operator in (6.77) requires only rI_n flops. Therefore, the overall cost of updating $\mathbf{U}_t^{(n)}$ in a naive way is $O\left(r \prod_{i=1}^{N-1} I_i\right)$. Note that $\Delta\widetilde{\mathbf{Y}}_t^{(n)}(\mathbf{V}_t^{(n)})^\top$ can be divided into two parts $\mathbf{Z}_t^{(n)} = \Delta\widetilde{\mathbf{Y}}_t^{(n)}\widetilde{\mathbf{W}}_t^{(n)}$ and $\mathbf{Z}_t^{(n)}(\mathbf{S}_t^{(n)})^{-\top}$. Here, $\Delta\widetilde{\mathbf{Y}}_t^{(n)}\widetilde{\mathbf{W}}_t^{(n)}$ can be referred to as "matricized tensor times Khatri-Rao product" (MTTKRP) [335, 336]. Fortunately, Phan *et al.* in [336] proposed a clever reorganization of MTTKRP which can accelerate the computation and reduce the overall cost of (6.77) to $O\left(r^2 \prod_{i=1,i\neq n}^{N-1} I_i\right)$.

Stage 3: Normalization and re-estimation of $\mathbf{u}_t^{(N)}$ (Optional)

In order to avoid numerical problems, we can perform the column-wise normalization on the updated factors $\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}$. In addition, given the already estimated factors, the weight vector $\mathbf{u}_t$ in Step 1 can be re-updated to achieve a better estimation as follows

$$\mathbf{u}_t^{(N)} = (\mathbf{H}_t^\top\mathbf{P}_t\mathbf{H}_t)^\#\mathbf{H}_t^\top\mathbf{P}_t\hat{\mathbf{y}}_t, \tag{6.79}$$

where $\mathbf{H}_t = \bigodot_{n=1}^{N-1}\mathbf{U}_t^{(n)}$. This step is useful for the early stage of tracking and fast time-varying environments [174, 211, 213].

---

[9]To enable the recursive rules of (6.75) and (6.77), $\mathbf{S}_{0,m}^{(n)}$ and $\mathbf{S}_0^{(n)}$ can be initialized by $\delta\mathbf{I}_r$ where $\delta > 0$, for $n = 1, 2, \ldots, N$.

6.3.2.2 Extensions of the RACP algorithm

In the following, we present two simple modifications of RACP when smoothness and nonnegativity are imposed on the loading factors.

Smoothness Condition

In many applications, smoothness is a common assumption under which the underlying data or model is supposed to be smooth [337]. Here, we incorporate a smoothing regularization matrix on the loading factors to control the smoothness of the solution as well as to avoid biases and singular/ill-posed computation. Particularly, this regularization adds a small bias against large terms into the updating rules.

On the arrival of $\boldsymbol{\mathcal{Y}}_t$, the outliers $\boldsymbol{O}_t$ and the coefficient vector $\mathbf{u}_t$ are derived from the following minimization:

$$
\begin{aligned}
\left\{\boldsymbol{O}_t, \mathbf{u}_t^{(N)}\right\} &= \operatorname*{argmin}_{\boldsymbol{O},\mathbf{u}} \; \|\boldsymbol{O}\|_1 + \frac{\gamma}{2}\|\mathbf{Bu}\|_2^2, \\
&\text{subject to } \left\|\boldsymbol{\mathcal{P}}_t \circledast \left(\boldsymbol{\mathcal{Y}}_t - \boldsymbol{O} - \boldsymbol{\mathcal{H}}_{t-1} \times_N \mathbf{u}\right)\right\|_F^2 = 0,
\end{aligned}
\tag{6.80}
$$

where $\boldsymbol{\mathcal{H}}_{t-1} = \boldsymbol{\mathcal{I}} \prod_{n=1}^{N-1} \times_n \mathbf{U}_{t-1}^{(n)}$ and $\gamma > 0$ is a small penalty parameter and $\mathbf{B}$ a chosen banded matrix. More concretely, the vector $\mathbf{u}_t$ is obtained by minimizing the following problem:

$$
\mathbf{u}_t^{(N)} = \operatorname*{argmin}_{\mathbf{u}} \; \frac{\gamma}{2}\|\mathbf{Bu}\|_2^2 + \frac{\rho}{2}\|\mathbf{P}_t(\mathbf{y}_t - \mathbf{o} - \mathbf{H}_{t-1}\mathbf{u})\|_2^2.
\tag{6.81}
$$

Accordingly, we replace the update rule for $\mathbf{u}$ in (6.56) with

$$
\mathbf{u}^i = \left(\mathbf{H}_{t-1}^\top \mathbf{P}_t \mathbf{H}_{t-1} + \frac{\gamma}{\rho}\mathbf{B}^\top\mathbf{B}\right)^{\#} \mathbf{H}_{t-1}^\top \mathbf{P}_t \left(\mathbf{y}_t - \mathbf{o}^i\right).
\tag{6.82}
$$

Instead of (6.65), the m-th row $\mathbf{u}_{t,m}^{(n)}$ of $\mathbf{U}_t^{(n)}$ is derived from

$$
\mathbf{u}_{t,m}^{(n)} = \operatorname*{argmin}_{\mathbf{u}_m^{(n)}} \sum_{\tau=t-L_t+1}^{t} \beta^{t-\tau} \left\|\underline{\mathbf{P}}_{\tau,m}^{(n)}\left(\left(\underline{\hat{\mathbf{y}}}_{\tau,m}^{(n)}\right)^\top - \mathbf{W}_\tau^{(n)}\left(\mathbf{u}_m^{(n)}\right)^\top\right)\right\|_2^2 + \frac{\gamma}{2}\left\|\mathbf{B}\left(\mathbf{u}_m^{(n)}\right)^\top\right\|_2^2,
\tag{6.83}
$$

In particular, $\mathbf{u}_{t,m}^{(n)}$ is the solution of the following equation:

$$
\sum_{\tau=t-L_t+1}^{t} \beta^{t-\tau}\left(\mathbf{W}_\tau^{(n)}\right)^\top \underline{\mathbf{P}}_{\tau,m}^{(n)}\left(\underline{\hat{\mathbf{y}}}_{\tau,m}^{(n)}\right)^\top
$$

$$
= \left(\sum_{\tau=t-L_t+1}^{t} \beta^{t-\tau}\left(\mathbf{W}_\tau^{(n)}\right)^\top \underline{\mathbf{P}}_{\tau,m}^{(n)}\mathbf{W}_\tau^{(n)} + \frac{\gamma}{2}\mathbf{B}^\top\mathbf{B}\right)\left(\mathbf{u}_m^{(n)}\right)^\top.
\tag{6.84}
$$

Therefore, the recursive rule of (6.75) becomes

$$\mathbf{u}_{t,m}^{(n)} = \mathbf{u}_{t-1,m}^{(n)} + \left(\boldsymbol{\delta}\underline{\tilde{\mathbf{y}}}_{-t,m}^{(n)}\right)^{\top}\left(\bar{\mathbf{V}}_{t,m}^{(n)}\right)^{\top}, \qquad (6.85)$$

where

$$\bar{\mathbf{V}}_{t,m}^{(n)} = \left(\mathbf{S}_{t,m}^{(n)} + \frac{\gamma}{2}\mathbf{B}^{\top}\mathbf{B}\right)^{-1}\left(\widetilde{\mathbf{W}}_{t}^{(n)}\right)^{\top}. \qquad (6.86)$$

Nonnegative Constraint

It is known that nonnegative tensor factorization (NTF) offers interesting properties, e.g., the resulting expression appears to be purely additive and the loading factors are "sparse" in general [338].

One of the simplest ways is to project the estimates (i.e., $\{\mathbf{U}_{t}^{(n)}\}_{n=1}^{N-1}$ and $\mathbf{u}_{t}^{(N)}$) on their nonnegative orthant at the end of each step of RACP, as introduced by Nguyen *et al.* in [174]. This approach offers a low complexity and yields a reasonable performance in some cases. However, it may not be optimal as well as guarantee the convergence in general. In this task, we aim to customize the updates of $\mathbf{u}_{t}^{(N)}$ and $\{\mathbf{U}_{t}^{(n)}\}_{n=1}^{N-1}$ for dealing with the nonnegativity at each time t.

In step 1, we particularly replace the exact LS solution (6.56) with the minimizer of the following nonnegative least-squares (NNLS) problem:

$$\mathbf{u}^{i} = \underset{\mathbf{u}}{\text{argmin}} \left\|\mathbf{P}_{t}\left(\mathbf{y}_{t} - \mathbf{o}^{i} - \mathbf{H}_{t-1}\mathbf{u}\right)\right\|_{2}^{2} \text{ subject to } [\mathbf{u}]_{j} \geq 0 \ \forall j. \qquad (6.87)$$

Here, we can apply any provable NNLS algorithm for solving (6.87), the reader is referred to [339, 340] for good surveys on numerical methods for NNLS. In this work, we adopt the widely-used algorithm of Lawson and Hanson [340] which is implemented as the function lsqnonneg in MATLAB.

In step 2, the m-th row of $\mathbf{U}_{t}^{(n)}$ can be derived from minimizing the following constrained version of (6.65):

$$\mathbf{u}_{t,m}^{(n)} = \underset{\mathbf{u}_{m}^{(n)}}{\text{argmin}} \sum_{\tau=t-L_{t}+1}^{t} \beta^{t-\tau} \left\|\underline{\mathbf{P}}_{-\tau,m}^{(n)}\left(\left(\hat{\underline{\mathbf{y}}}_{-\tau,m}^{(n)}\right)^{\top} - \mathbf{W}_{\tau}^{(n)}\left(\mathbf{u}_{m}^{(n)}\right)^{\top}\right)\right\|_{2}^{2},$$

$$\text{subject to } \left[\mathbf{u}_{m}^{(n)}\right]_{j} \geq 0 \ \forall j. \qquad (6.88)$$

To solve (6.88), we apply the projected gradient method (i.e., proximal gradient on indicator function [118]). More concretely, the iterative procedure for

updating $\mathbf{u}_{t,m}^{(n)}$ is given by[10]

$$\mathbf{u}_j = \left[\left(\mathbf{I}_r - \frac{\mathbf{S}_{t,m}^{(n)}}{\left\| \mathbf{S}_{t,m}^{(n)} \right\|_2} \right) \mathbf{u}_{j-1} - \frac{\mathbf{d}_{t,m}^{(n)}}{\left\| \mathbf{S}_{t,m}^{(n)} \right\|_2} \right]_+ , \tag{6.89}$$

where j denotes the iteration index. We refer to this modification of RACP as NRACP.

6.3.3 Performance Analysis

In this section, we present a theoretical convergence analysis for the proposed RACP method in Algorithm 1 while assuming $\mathbf{D}_t = \mathbf{D}$ is fixed. Inspired by the recent results of our companion works on robust subspace tracking [25] and tensor tracking [30], we establish a unified theoretical approach to analyse the convergence of the objective values $\{f_t(\mathbf{D}_t)\}_{t=1}^{\infty}$ as well as the solutions $\{\mathbf{D}_t\}_{t=1}^{\infty}$ generated by RACP.

6.3.3.1 Assumptions

In order to facilitate the convergence analysis, we make the following assumptions:[11]

(A1) Low-rank components $\{\boldsymbol{\mathcal{Y}}_t\}_{t\geq 1}$ of the observed tensor slices $\{\boldsymbol{\mathcal{Y}}_t\}_{t\geq 1}$ are supposed to be deterministic and bounded. Entries of noise tensors $\{\boldsymbol{\mathcal{N}}_t\}_{t\geq 1}$ are zero-mean, independently and identically distributed (i.i.d.) with a small finite covariance, and bounded. Entries of $\boldsymbol{\mathcal{Y}}_t$ are Frobenius-norm bounded, i.e., $\|\boldsymbol{\mathcal{Y}}_t\|_F \leq M_x < \infty$, for all t.

(A2) The dictionary $\mathbf{D}_t$ remains unchanged over time (i.e., $\mathbf{D}_t = \mathbf{D}$). The loading factors are Frobenius-norm bounded and the tensor rank r is fixed.

(A3) Observation masks $\{\boldsymbol{\mathcal{P}}_t\}_{t\geq 1}$ are independent of $\{\boldsymbol{\mathcal{Y}}_t\}_{t\geq 1}$, and their entries follow a uniform distribution. The number of observed entries of $\boldsymbol{\mathcal{Y}}_t$ should be larger than the lower bound $O(rL\log(L))$, where $L = I_1 I_2 \dots I_N$. Every row of the mode-n unfolding $\underline{\mathbf{Y}}_t^{(n)}$ of $\boldsymbol{\mathcal{Y}}_t$ is observed in at least r entries, for $n = 1, 2, \dots, N$. In addition, each observed entry

---

[10]Projected gradient descent has a form of $\mathbf{u}_j = \left[\mathbf{u}_{j-1} - \eta_j \nabla \tilde{f}_t(\mathbf{u}_{j-1}) \right]_+$, where $\nabla \tilde{f}_t(\mathbf{u}_m^{(n)}) = \mathbf{S}_{t,m}^{(n)} \mathbf{u}_m^{(n)} - \mathbf{d}_{t,m}^{(n)}$. In practice, we can set the value of the step-size η_j to $1/\mathcal{L}$ where $\mathcal{L}$ is the Lipschitz constant of $\nabla \tilde{f}_t(\mathbf{u}_m^{(n)})$. In this work, it is easy to indicate that $\mathcal{L} = \left\| \mathbf{S}_{t,m}^{(n)} \right\|_2$.

[11]The four assumptions (A1)-(A4) are used for the purpose of convergence analysis only, the proposed RACP algorithm can work well in many other scenarios, please see Sec. V for details

of $\boldsymbol{\mathcal{Y}}_t$ is corrupted by outliers independently of others, i.e., the index of outliers is also uniformly random.

(A4) $\tilde{f}_t(\cdot)$ is m-strongly multi-block convex, i.e., its second-order derivative w.r.t. each factor is positive-definite, $\nabla_n^2 \tilde{f}_t(\mathbf{U}^{(n)}, .) \succeq m\mathbf{I} \succ \mathbf{0}$ with $m > 0$.

Among them, assumptions (A1) and (A2) are common for analysing the convergence of online learning algorithms, such as [25, 106, 120]. Indeed, (A1) holds in many situations, e.g., real data are often bounded such as audio, image and video. (A2) is a strong assumption as it requires the tensor dictionary to be constant with time. It also prevents arbitrarily large values in $\mathbf{U}^{(n)}$ and ill-conditioned computation. Along with (A1), it is interpreted as the simplest possible data model in (robust) tensor tracking where tensor slices are supposed to be generated from a stationary process. Theoretically, stationary processes are often "easier" to model and analyse than nonstationary ones as their statistical properties remain constant over time. Accordingly, stationary has become a common assumption underlying many statistical procedures in general and tracking tools in particular to study their convergence and asymptotic behavior. In this work, a novel theoretical approach is established to analyse the convergence behavior of RACP in stationary environments. We leave the convergence analysis of RACP under a nonstationary model where the tensor dictionary is time-varying to a future work. Assumption (A3) is also common, under which the index of missing entries is uniformly random. Moreover, with respect to the imputation of missing values and recovery of low-rank components, the uniform randomness allows the sequence of binary masks $\{\boldsymbol{\mathcal{P}}_t\}_{t \geq 1}$ to admit stable recovery [319]. The next two constraints of (A3) are fundamental conditions to prevent the underdetermined imputation problem [341, 342]. The last constraint of (A3) plays a similar role as the first one but accounting for sparse outliers. Assumption (A4) allows us to derive several nice results in the convergence analysis. In fact, the Hessian matrix of $\tilde{f}_t(\cdot)$ w.r.t. each factor is already positive semidefinite, (A4) can be achieved with a good initialization $\mathbf{D}_0$ or by simply adding a convex regularization term into $\tilde{\ell}(\cdot)$ or $\tilde{f}_t(\cdot)$.

6.3.3.2 Main Results

Given the assumptions of (A1)-(A4), our main theoretical result can be stated in the following theorem:

Theorem 4 *Given (A1)-(A4), $L_t = t$ and let $\mathbf{D}_t$ be the solution generated by Algorithm 1 at each time t. When $t \to \infty$,*

- $f_t(\mathbf{D}_t) - \tilde{f}_t(\mathbf{D}_t) \overset{a.s.}{\to} 0$;

- $\nabla f_t(\mathbf{D}_t) \overset{a.s.}{\to} 0$.

Accordingly, $\mathbf{D}_t$ is almost surely a stationary point of $f_t(.)$ when t tends to infinity.

The proof of this theorem follows intermediately Proposition 11 and Lemmas 12 and 13, to be stated shortly. We detail their proofs in our appendix.

Lemma 11 (Key Properties) *Given (A1)-(A4), $L_t = t$, and denote the error function $e_t := \tilde{f}_t - f_t$. If $\{\mathbf{D}_t, \boldsymbol{O}_t, \mathbf{u}_t\}_{t=1}^{\infty}$ is a sequence of variables generated by Algorithm 1, then*

(a) *Boundedness: $\{\mathbf{D}_t, \boldsymbol{O}_t, \mathbf{u}_t\}_{t=1}^{\infty}$ are uniformly bounded;*

(b) *Forward Monotonicity: $\tilde{f}_t(\mathbf{D}_{t-1}) \geq \tilde{f}_t(\mathbf{D}_t)$;*

(c) *Backward Monotonicity: $\tilde{f}_{t-1}(\mathbf{D}_{t-1}) \leq \tilde{f}_{t-1}(\mathbf{D}_t)$;*

(d) *Stability of Estimates: $\|\mathbf{D}_t - \mathbf{D}_{t-1}\|_F = \mathcal{O}(1/t)$;*

(e) *Stability of Errors: $|e_t(\mathbf{D}_t) - e_{t-1}(\mathbf{D}_{t-1})| = \mathcal{O}(1/t)$.*

Proof Sketch. Part (a) can be derived from applying the same arguments of Proposition 11 in our companion work [30]. Parts (b) and (c) are trivial due to the proposed BCD scheme. Part (d) can be obtained by exploiting the Lipschitz continuity and multi-block convexity of the surrogate function $\tilde{f}_t$. We indicate Part (e) by using Part (d) and the Lipschitz continuity of f and $\tilde{f}$.

Lemma 12 (Almost sure convergence) *The sequence of $\{\tilde{f}_t(\mathbf{D}_t)\}_{t=1}^{\infty}$ converges almost surely as $t \to \infty$. The sequence of objective values $\{f_t(\mathbf{D}_t)\}_{t=1}^{\infty}$ converges to the same limit of its surrogate $\{\tilde{f}_t(\mathbf{D}_t)\}_{t=1}^{\infty}$, i.e.,*

$$f_t(\mathbf{D}_t) \to \tilde{f}_t(\mathbf{D}_t) \quad a.s. \tag{6.90}$$

Proof Sketch. We first prove that

$$\sum_{t=1}^{\infty} \mathbb{E}\left[\delta_t \mathbb{E}\left[\tilde{f}_{t+1}(\mathbf{D}_{t+1}) - \tilde{f}_t(\mathbf{D}_t)\big|\mathcal{F}_t\right]\right] < \infty, \tag{6.91}$$

where $\mathcal{F}_t = \{\mathbf{D}_\tau, \boldsymbol{O}_\tau, \mathbf{u}_\tau\}_{0<\tau\leq t}$ records all past estimates of RACP at time t and the indicator function δ_t is defined as

$$\delta_t \triangleq \begin{cases} 1 & \text{if} \ \ \mathbb{E}\big[\tilde{f}_{t+1}(\mathbf{D}_{t+1}) - \tilde{f}_t(\mathbf{D}_t)\big|\mathcal{F}_t\big] > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6.92}$$

Thanks to the quasi-martingale convergence theorem [343, page 51], (6.91) implies that $\{\tilde{f}_t(\mathbf{D}_t)\}_{t=1}^\infty$ converges almost surely as $t \to \infty$.

We next prove $\{f_t(\mathbf{D}_t)\}_{t=1}^\infty$ and $\{\tilde{f}_t(\mathbf{D}_t)\}_{t=1}^\infty$ converge to the same limit by showing

$$\sum_{t=1}^\infty \frac{\tilde{f}_t(\mathbf{D}_t) - f_t(\mathbf{D}_t)}{t + 1} < \infty. \tag{6.93}$$

Since $\sum_{t=1}^\infty \frac{1}{t+1} = \infty$ and $\big|e_t(\mathbf{D}_t) - e_{t-1}(\mathbf{D}_{t-1})\big| = \mathcal{O}(1/t)$, we obtain $\sum_{t=1}^\infty \tilde{f}_t(\mathbf{D}_t) - f_t(\mathbf{D}_t) < \infty$, or

$$\tilde{f}_t(\mathbf{D}_t) \to f_t(\mathbf{D}_t) \ a.s., \tag{6.94}$$

thanks to [120, Lemma 3].

Lemma 13 (Local convergence) *When $t \to \infty$, $\mathbf{D}_t$ converges almost surely to a stationary point of $\tilde{f}_\infty(.) = \lim_{t\to\infty} \tilde{f}_t(.)$:*

$$\nabla \tilde{f}_\infty(\mathbf{D}_t) \to \nabla f_\infty(\mathbf{D}_t) \to 0 \ \ a.s. \tag{6.95}$$

Proof Sketch. We first indicate that

$$\lim_{t\to\infty} \operatorname{tr}\Big[(\mathbf{D}_t - \mathbf{D}_{t+1})^\top \nabla \tilde{f}_{t+1}(\mathbf{D}_{t+1})\Big] = 0, \tag{6.96}$$

by showing $\sum_{t=1}^\infty \Big|\operatorname{tr}\big[(\mathbf{D}_t - \mathbf{D}_{t+1})^\top \nabla \tilde{f}_{t+1}(\mathbf{D}_{t+1})\big]\Big| < \infty$.

Next, we prove that the following inequality

$$\operatorname{tr}\Big[(\mathbf{D}_t - \mathbf{D}_{t+1})^\top \nabla \tilde{f}_{t+1}(\mathbf{D}_{t+1})\Big] \leq c_1 \big\|\mathbf{D}_{t+1} - \mathbf{D}_t\big\|_F^2$$
$$+ c_2 \operatorname{tr}\Big[(\mathbf{D} - \mathbf{D}_t)^\top \nabla \tilde{f}_{t+1}(\mathbf{D}_t)\Big], \tag{6.97}$$

holds for all $\mathbf{D} \in \mathcal{D}$ where c_1 and c_2 are positive constants.

Then, we use proof by contradiction to indicate that

$$\big(\nabla \tilde{f}_\infty(\mathbf{D}_\infty)\big)^\top (\mathbf{D} - \mathbf{D}_\infty) \geq 0, \ \forall \mathbf{D} \in \mathcal{D}. \tag{6.98}$$

Accordingly, $\mathbf{D}_\infty$ is a stationary point of $\tilde{f}_\infty(.)$.

In order to prove $\nabla \tilde{f}_t(\mathbf{D}_t) \overset{a.s.}{\to} \nabla f_t(\mathbf{D}_t)$ as $t \to \infty$, we first exploit that $f_t(\mathbf{D}+a_t\mathbf{V}) \leq \tilde{f}_t(\mathbf{D}+a_t\mathbf{V}) \; \forall \mathbf{D}, \mathbf{V} \in \mathcal{D}$ and a_t, and then take its Taylor expansion at $t \to \infty$ to yield

$$f_\infty(\mathbf{D}_\infty) + \mathrm{tr}\left[a_t\mathbf{V}^\top \nabla f_\infty(\mathbf{D}_\infty)\right] + o(a_t\mathbf{V})$$
$$\leq \tilde{f}_\infty(\mathbf{D}_\infty) + \mathrm{tr}\left[a_t\mathbf{V}^\top \nabla \tilde{f}_\infty(\mathbf{D}_\infty)\right] + o(a_t\mathbf{V}). \tag{6.99}$$

As indicated in Lemma 12, $\tilde{f}_\infty(\mathbf{D}_\infty) = f_\infty(\mathbf{D}_\infty)$ and thus

$$\mathrm{tr}\left[a_t\mathbf{V}^\top \nabla f_\infty(\mathbf{D}_\infty)\right] \leq \mathrm{tr}\left[a_t\mathbf{V}^\top \nabla \tilde{f}_\infty(\mathbf{D}_\infty)\right].$$

Since the above inequality must hold for all $\mathbf{V} \in \mathcal{D}$ and a_t, we obtain

$$\nabla \tilde{f}_\infty(\mathbf{D}_\infty) = \nabla f_\infty(\mathbf{D}_\infty). \tag{6.100}$$

Together with (6.98), we can conclude that $\mathbf{D}_\infty$ is a stationary point of the objective function $f_t(.)$ as $t \to \infty$.

6.3.3.3   Discussions

Our analysis follows the same framework to derive the convergence of adaptive/incremental algorithms for online matrix/tensor factorization problems as in [25, 30, 106, 120, 121, 176]. Therefore, our main theoretical result is somehow similar to their results. However, there are several points that make our convergence analysis different from theirs.

First, [120] is devoted to the problem of online dictionary learning and sparse coding. The authors dealt with a LASSO-like cost function and required a preliminary uniqueness condition on the sparse coding. The condition is important to ensure that the solution generated in the sparse coding stage is unique, and to derive the Lipschitz property of the cost function. Particularly, they suggested an elastic-net regularized term for enforcing the condition. Since the problem formulation of RTT is different, our convergence analysis does not involve such issues. Moreover, the missing data distinguishes our work from theirs.

The studies in [121] and [25] consider the problem of robust online PCA and/or subspace tracking which can handle data corruptions (i.e., outliers and/or missing entries). These studies are designed for tracking the time-variant subspace – an object different from ours – which leads to some differences from our analysis. In particular, their main goal is to develop provable algorithms for minimizing the expected cost function in an online manner, and then indicate that their algorithm converges to a stationary point or global optimum under certain conditions. Our optimization, however, minimizes an exponential weighted cost function constructed on the latest data

streams (i.e., tensor slices). Moreover, [121] does not require the solution derived from the subspace update stage necessarily optimal, but full column rank only at each time t (see [121, Theorem 1]). However, it is a sufficient condition on which we highly leverage in our analysis. In addition, our object is a set of multiple loading factors, instead of a single subspace matrix as in [25, 121].

The studies most related to ours are those in [30, 106, 176], which also investigate the tensor tracking problem. However, they consider only outlier-free streaming tensors. By contrast, we here provide a more unified convergence analysis that is able to deal with both missing data and outliers. Also, our results are stronger than those of [106, 176], being limited to the case of third-order streaming tensors with $\beta = 1$.

6.4   Performance Evaluation

In this section, we provide several experiments on both synthetic and real data to demonstrate the effectiveness of the proposed algorithms, ACP, ATD, and RACP. We also compare them with several state-of-the-art algorithms to provide practical evidences of their effectiveness and efficiency. All experiments are implemented on MATLAB a windows computer with an Intel Core i5-8300H and 16GB of RAM.[12]

6.4.1   Performance of ACP

We assess the performance of ACP w.r.t. the following aspects: (i) impact of algorithm parameters on its tracking ability; (ii) performance of ACP in non-stationary and time-varying environments; (iii) effectiveness and efficiency of ACP as compared with other adaptive CP algorithms.

6.4.1.1   Experiment Setup

According to the setup of OLSTEC [176], a time-varying model for streaming tensors is constructed as follows.

At $t = 0$, the loading factor $\mathbf{U}_t^{(n)}$ is generated at random whose entries are i.i.d. drawn from the Gaussian distribution $\mathcal{N}(0, 1)$. At time $t > 0$, $\mathbf{U}_t^{(n)} \in \mathbb{R}^{I_n \times r}$ is varied under the model

$$\mathbf{U}_t^{(n)} = \mathbf{U}_{t-1}^{(n)} \mathbf{Q}_t, \tag{6.101}$$

---

[12]Our codes are available at: `https://github.com/thanhtbt/tensor_tracking/`.

where $\mathbf{Q}_t \in \mathbb{R}^{r \times r}$ is a rotation matrix to control the variation of $\mathbf{U}^{(n)}$ between instances t and $t - 1$, which is defined by

$$
\mathbf{Q}_t = \left[\begin{array}{c|ccc|c} \mathbf{I}_{p_t-1} & 0 & 0 & 0 \\ \hline 0 & \cos(\alpha_t) & -\sin(\alpha_t) & 0 \\ 0 & \sin(\alpha_t) & \cos(\alpha_t) & 0 \\ \hline 0 & 0 & 0 & \mathbf{I}_{r-p_t-1} \end{array} \right],
\tag{6.102}
$$

where $p_t = \mathrm{mod}(t + r - 2, r - 1) + 1$ and α_t is the rotation angle. Specifically, the higher value of α_t is, the faster the loading factor $\mathbf{U}^{(n)}$ changes.

The t-th slice $\boldsymbol{\mathcal{Y}}_t$ with missing entries is then derived from the following model:

$$
\boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{P}}_t \circledast \left(\left[\!\left[\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}, \mathbf{u}_t^{(N)} \right]\!\right] + \sigma \boldsymbol{\mathcal{N}}_t \right),
\tag{6.103}
$$

where $\boldsymbol{\mathcal{P}}_t$ is a binary mask tensor whose entries are generated randomly using the Bernoulli model with the probability ρ, i.e., ρ represents the missing density in the measurement; $\boldsymbol{\mathcal{N}}_t$ is a Gaussian noise tensor (with zero-mean, unit power entries) of the same size of $\boldsymbol{\mathcal{Y}}_t$ and the factor σ is to control the noise level; and the weight vector $\mathbf{u}_t^{(N)}$ is a Gaussian random vector living on $\mathbb{R}^r$ space.

In order to evaluate estimation accuracy, we measure the relative error (RE) metric defined by

$$
\mathrm{RE}(\mathbf{A}_{tr}, \mathbf{A}_{es}) = \frac{\|\mathbf{A}_{tr} - \mathbf{A}_{es}\|_F}{\|\mathbf{A}_{tr}\|_F},
\tag{6.104}
$$

where $\mathbf{A}_{tr}$(resp. $\mathbf{A}_{es}$) refers to the ground truth (resp. estimation)[13].

6.4.1.2 Effect of Forgetting Factor β

The choice of β plays a central role in how effective and efficient ACP can be in nonstationary environments. In order to investigate the effect of the forgetting factor, we vary the value of β from 0 to 1 and measure estimation accuracy of ACP in different tests with regard to the rotational angle α. Fig. 6.3 illustrates the experimental results of applying ACP to a synthetic 4-order tensor whose size is $20 \times 20 \times 20 \times 500$ and its rank $r = 5$. The noise level σ is set at 10^{-3}, while the sketching parameter m is fixed at 10. It is clear that

---

[13]Due to the permutation and scaling indeterminacy of the CP decomposition, we can find $\mathbf{U}_{es}$ which is matched with $\mathbf{U}_{tr}$ from $\mathbf{U}_t$, as follows: $\mathbf{U}_{es} = \mathbf{U}_t \mathbf{P}^\top \mathbf{D}^{-1}$, where the permutation matrix $\mathbf{P} \in \mathbb{R}^{r \times r}$ and the diagonal matrix $\mathbf{D} \in \mathbb{R}^{r \times r}$ are derived from minimizing the optimization $\mathrm{argmin}_{\mathbf{D},\mathbf{P}} \|\mathbf{U}_t - \mathbf{U}_{tr}\mathbf{D}\mathbf{P}\|_F^2$.
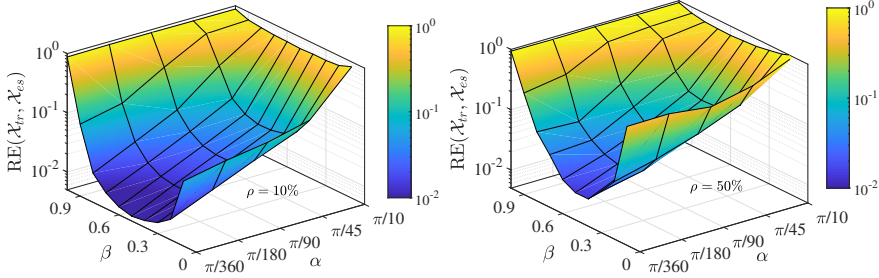
Figure 6.3: Effect of the forgetting factor β on the performance of ACP versus the rotation angle α.

(a) $\sigma = 0$

(b) $\beta = 1$
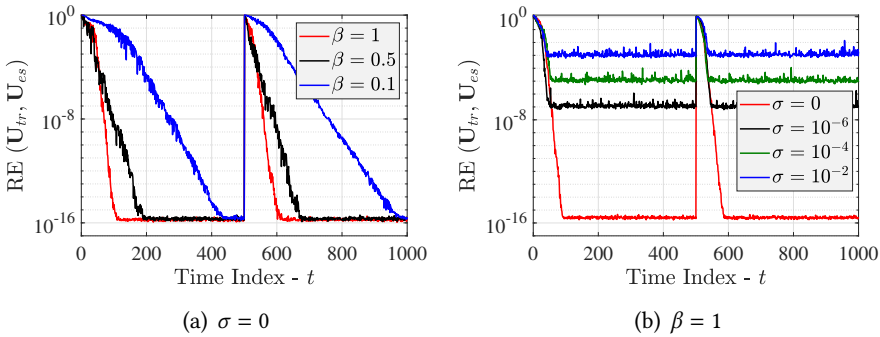
Figure 6.4: Performance of ACP in stationary environments: $\mathcal{Y}_t \in \mathbb{R}^{20 \times 20 \times 20 \times 1000}$, the true rank $r = 5$, an abrupt change at $t = 500$.

the optimal value of β depends not only on the rotation angle α, but also on the missing density ρ. When β increases from 0 to 1, the performance of ACP goes up first and then drops. As can be seen in Fig. 6.3 that the value of β should be around 0.5 for reasonable performance. Thus, we fix $\beta = 0.5$ in the next experiments for. It is worth noting that in stationary environments, we can set the value of $\beta = 1$ to achieve the best performance, please see Fig. 6.4 for an illustration.

6.4.1.3   Asymptotic Convergence Behavior

We next illustrate the convergence behavior of ACP in terms of the variation $\|\mathbf{U}_{t+1} - \mathbf{U}_t\|_F$ and the objective value $f_t(\mathcal{U}_t)$. We use the same 4-order tensor above but with 1000 tensor slices. Two noise levels are considered (including $\sigma = 0$ and $\sigma = 10^{-3}$), while the missing density ρ is chosen among $\{10\%, 30\%, 50\%\}$. The experiment results are shown as in Fig. 6.5. We can see that convergence results agree with those stated in the proof sketch of Lemma 9.

(a) $f_t(\boldsymbol{\mathcal{U}}_t)$: Noise-free

(b) $\|\mathbf{U}_{t+1} - \mathbf{U}\|_F$: Noise-free

(c) $f_t(\boldsymbol{\mathcal{U}}_t)$: $\sigma = 10^{-3}$

(d) $\|\mathbf{U}_{t+1} - \mathbf{U}\|_F$: $\sigma = 10^{-3}$
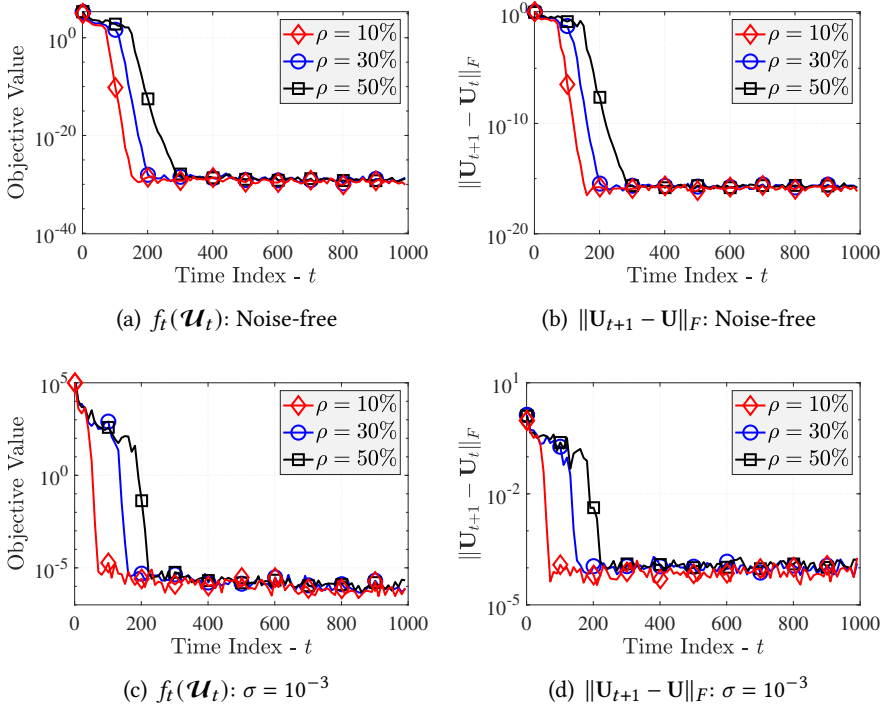
Figure 6.5: Convergence behavior of ACP in terms of the objective values $f_t(\boldsymbol{\mathcal{U}}_t)$ **and** $\|\mathbf{U}_{t+1} - \mathbf{U}_t\|_F$.

6.4.1.4 Noisy and Dynamic Environments

First, the robustness of ACP is investigated against the noise variance. We test ACP's tracking ability on the same static 4-order tensor above with different values of the noise level σ. Fig. 6.6 shows that the value of σ does not affect the convergence rate of ACP, but only its estimation error. Specifically, when we increase the noise level σ, the relative error (RE) between the ground truth and estimation goes up gradually, but towards an error bound.

Next, we use the same tensor, but the number of slices is double for illustrating the robustness of ACP against time-varying environments. In particular, the proposed algorithm is evaluated in two scenarios, including a slow time-varying case (i.e., $\alpha = \pi/360$) and a fast time-varying case (i.e., $\alpha = \pi/45$). Also, at time $t = 600$, we make an abrupt change in these models. In addition, the missing density ρ is chosen among $\{10\%, 30\%, 50\%\}$.

Experimental results indicate that ACP is capable of tracking streaming tensors in dynamic environments, as shown in Fig. 6.7. In both scenarios, the relative error (RE) between the ground truth and estimation always converges towards a steady state error bound. The missing density ρ has only an
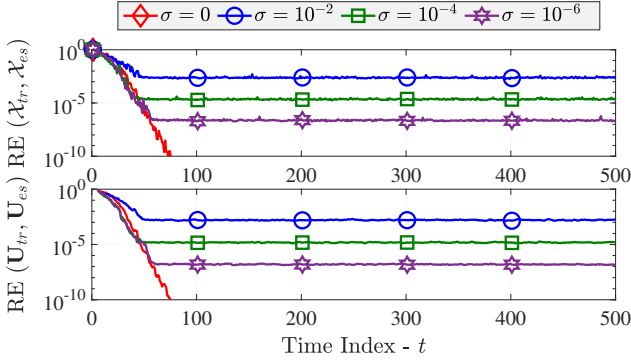
Figure 6.6: Effect of the noise level σ on the performance of ACP.
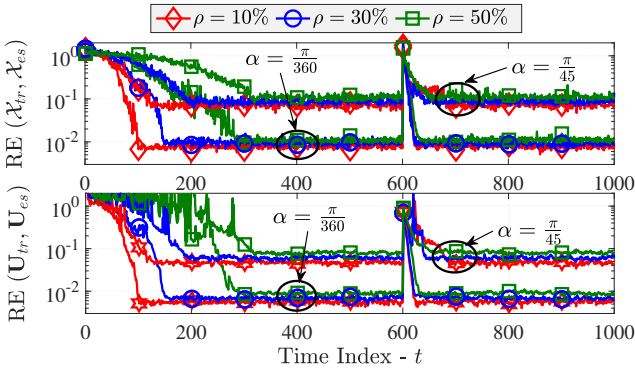
Figure 6.7: Time-varying scenarios: ACP's tracking ability versus the missing density ρ and the rotation angle α: The noise level $\sigma = 10^{-3}$ and an abrupt change at $t = 600$.

influence on the convergence rate of ACP. Specifically, the lower the missing density ρ is, the faster ACP converges.

6.4.1.5   Evaluation of Effectiveness and Efficiency

To demonstrate the effectiveness and efficiency of our algorithm, we compare performance of ACP in terms of estimation accuracy and running time with the state-of-the-art adaptive CP decompositions for incomplete tensors, including OLSTEC [176], CP-PETRELS [215], TeCPSGD [106]. For a fair comparison, parameters of these algorithms are fine-tuned carefully to achieve good performance. Particularly, the forgetting factor λ is set at 0.7, 0.001, and 0.98, respectively, for OLSTEC, TeCPSGD and CP-PETRELS. Moreover, OLSTEC and TeCPSGD are also dependent on a regularization parameter μ
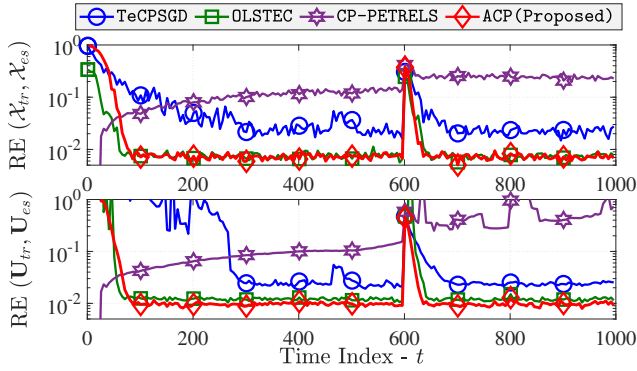
Figure 6.8: Tracking ability of four adaptive CP algorithms in a time-varying scenario with 50% **missing observations: The tensor of size** $20 \times 20 \times 1000$**, the noise level** $\sigma = 10^{-3}$**, the rotation angle** $\alpha = \pi/360$ **and an abrupt change at** $t = 600$.
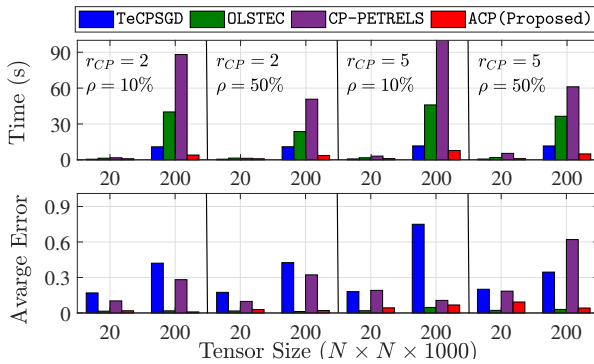
Figure 6.9: Performance of four adaptive CP algorithms on synthetic 3-order tensors: The noise level $\sigma = 10^{-3}$ **and the rotation angle** $\alpha = \pi/360$**.**

which is set at 10^{-3} and 10^{-1} respectively.

Since these algorithms are capable of tracking 3-order tensors only, we use synthetic streaming tensors of size $N \times N \times 1000$ in this task. The noise level is fixed at $\sigma = 10^{-3}$. Performance of these algorithms is evaluated on a small tensor $20 \times 20 \times 1000$ and a big tensor $200 \times 200 \times 1000$. Results are shown in Figs. 6.8 and 6.9. We can see that OLSTEC and ACP provide comparative estimation accuracy. In terms of running time, ACP is several times faster than OLSTEC, especially in big tensor tests. TeCPSGD is a fast adaptive algorithm, but yields lower estimation accuracy as compared to ACP and OLSTEC, while CP-PETRELS gives the worst accuracy as well as running time.

6.4.2   Performance of ATD

The following experiments will evaluate the ability of ATD for the problem of tensor tracking.

6.4.2.1   Experimental Setup

Follow the setup above, the incomplete slice $\boldsymbol{\mathcal{Y}}_t$ at time t is generated randomly using the following model:

$$\boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{P}}_t \circledast \left(\left[\!\!\left[\boldsymbol{\mathcal{G}}_t; \{\mathbf{U}_t^{(n)}\}_{t=1}^{N-1}, \mathbf{u}_t^{(N)} \right]\!\!\right] + \sigma \boldsymbol{\mathcal{N}}_t \right), \qquad (6.105)$$

where the loading factor $\mathbf{U}_t^{(n)}$ and the core tensor $\boldsymbol{\mathcal{G}}_t$ are updated by the following rules

$$\mathbf{U}_t^{(n)} = \mathbf{U}_{t-1}^{(n)} + \varepsilon \mathbf{N}_t^{(n)} \text{ and } \boldsymbol{\mathcal{G}}_t = \boldsymbol{\mathcal{G}}_{t-1} + \varepsilon \boldsymbol{\mathcal{V}}_t, \qquad (6.106)$$

where $\mathbf{U}_0^{(n)}, \mathbf{N}_t^{(n)} \in \mathbb{R}^{I_n \times r_n}$ and $\boldsymbol{\mathcal{V}}_t \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_N}$ are the Gaussian noises whose entries are distributed i.i.d from $\mathcal{N}(0, 1)$ and the time-varying factor ε is to control their variation.

   Besides the relative error (RE) metric, we also use the subspace estimation performance (SEP) [62] metric to evaluate the subspace estimation accuracy, which is defined by

$$\text{SEP}(\mathbf{U}_{tr}, \mathbf{U}_{es}) = \frac{\text{tr}\left(\mathbf{U}_{es}^{\#}(\mathbf{I} - \mathbf{U}_{tr}\mathbf{U}_{tr}^{\#})\mathbf{U}_{es}\right)}{\text{tr}\left(\mathbf{U}_{es}^{\#}(\mathbf{U}_{tr}\mathbf{U}_{tr}^{\#})\mathbf{U}_{es}\right)}, \qquad (6.107)$$

where $\mathbf{U}_{tr}$(resp. $\mathbf{U}_{es}$) refers to the true loading factor (resp. estimated factor). The lower value of SEP is, the better accuracy the algorithm achieves.

6.4.2.2   Robustness of ATD

In order to demonstrate the robustness of ATD against data corruption, we change the number of missing entries in the measurement by varying the value of ρ and evaluate its performance on different noise levels. We also compare ATD with three well-known batch Tucker algorithms for tensor completion, including iHOOI [344], ALSaS [344], and WTucker [345]. These algorithms are iterative-based, so their procedure will be stopped when the accuracy tolerance tol or the maximum iteration ITER_{max} has been met. For convergence guarantee, we fix the value of tol at 10^{-4}, while the value of ITER_{max} is set at 500, 500, and 100, respectively, for iHOOI, ALSaS and WTucker. For ATD, the forgetting factor λ is fixed at 0.7 in the following experiments.
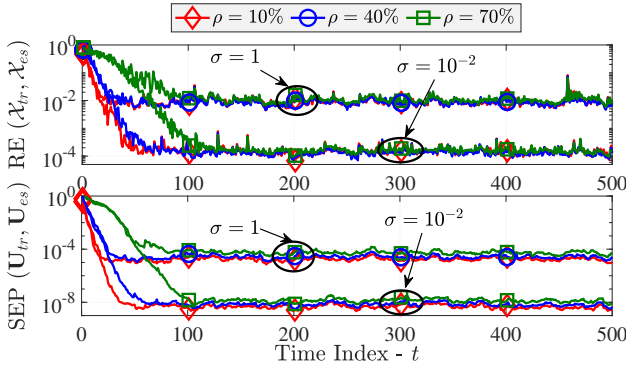
Figure 6.10: Performance of ATD versus the missing density ρ and the noise level σ: On the 4**-order tensor of size** $20 \times 20 \times 20 \times 500$ **and its Tucker rank** $\mathbf{r}_{\text{TD}} = [3, 3, 3, 3]$.

In this task, we use a static tensor of size $20 \times 20 \times 20 \times 500$ (i.e., the time-varying factor $\varepsilon = 0$) whose core is generated at random from a Gaussian distribution of zero-mean and unit variance. Under the Tucker model with $\mathbf{r}_{\text{TD}} = [3, 3, 3, 3]$, performance of ATD on the tensor is illustrated in Fig. 6.10. Results show that ATD can successfully track the multilinear low-rank model in all test cases. Similar to ACP, the missing density ρ has influence only on the convergence rate of ATD, i.e., the higher the value of ρ is, the slower ATD converges. Performance comparison among Tucker algorithms is reported statistically in Tab. 6.1 and shown in Fig. 6.11. Results indicate that ATD is the fastest algorithm, much faster than the state-of-the-art algorithms. For instance, when dealing with the case of 50% missing observations and $\mathbf{r}_{\text{TD}} = [3, 3, 3, 3]$, the running time of ATD is only 2.51 seconds and 35 times faster than the second-fastest algorithm, iHOOI. Moreover, ATD always provides good estimation accuracy in terms of both SEP metric and RE metric as compared to that of the batch algorithms.

6.4.2.3 Tracking Ability in Dynamic Environments

We continue to investigate the tracking ability of ATD in nonstationary and time-varying environments by changing the time-varying factor ε in the range $[10^{-4}, 10^{-1}]$. We use the same tensor dimensions as in the previous task. Also, we create a significant subspace change at time $t = 300$ to see how fast ATD can converge. Fig. 6.12 shows the convergence behavior of ATD versus the time-varying factor ε. We can see that the convergence rate of ATD is not affected by ε but only its estimation error.

Table 6.1: Performance of Tucker algorithms on a static 4**-order tensor of size** $20 \times 20 \times 20 \times 500$ **and the noise level** $\sigma = 10^{-2}$**.**

Rank	[3, 3, 3, 3]						[10, 10, 10, 10]					
Missing	$\rho = 50\%$			$\rho = 70\%$			$\rho = 50\%$			$\rho = 70\%$		
Metric	RE($\mathcal{X}$)	SEP(U)	Time(s)	RE($\mathcal{X}$)	SEP(U)	Time(s)	RE($\mathcal{X}$)	SEP(U)	Time(s)	RE($\mathcal{X}$)	SEP(U)	Time(s)
iHOOI	3.0e-4	4.2e-8	88.1	8.1e-4	4.7e-7	345.3	9.1e-2	5.1e-4	192.9	3.5e-1	1.3e-2	571.5
ALSaS	3.1e-4	4.3e-8	109.9	7.8e-4	4.9e-7	539.5	1.0e-4	2.8e-9	719.1	8.3e-4	3.4e-8	3754.6
WTucker	2.1e-4	2.4e-8	209.1	3.5e-4	1.3e-7	597.4	3.7e-5	2.8e-10	241.2	5.0e-5	3.3e-10	631.7
ATD	**6.4e-5**	**7.6e-9**	**2.5**	**1.8e-4**	**1.4e-8**	**5.7**	**1.7e-5**	**6.8e-11**	**21.7**	**3.2e-5**	**2.5e-10**	**58.2**

6.4.2.4 Orthogonality Constraint

In practice, Tucker decomposition is often considered with orthogonality constraints on the loading factors. The unconstrained ATD can be recast into an orthogonal ATD while retaining the equivalent approximation error. To demonstrate this point, we set up a time-varying scenario and compare the performance of ATD and ATD with the orthogonalization step, called ATD-O. Fig. 6.12 indicates that the convergence rate of ATD-O is slightly better than that of the unconstrained ATD, but both yield the same error floor. Due to space limitation, we here omit results with ATD-O and presents only those of ATD.

6.4.2.5 Real Data

To demonstrate the effectiveness of our algorithms on real datasets, we consider two related applications: video completion and multichannel EEG analysis.

Video Completion. In this task, four real video surveillance sequences are used, including `Highway`, `Hall`, `Lobby` and `Park`[14]. Specifically, `Highway` contains 1700 frames of size 320×240 pixels. `Hall` has 3584 frames of size 174×144 pixels. `Lobby` consists of 1546 frames of size 128×160 pixels. `Park` includes 600 frames of size 288×352 pixels.

We first investigate the effect of the forgetting factor λ on the reconstruction performance of the two proposed algorithms for video completion. Particularly, the value of λ and the missing ratio ρ are varied from 0.1 to 0.9. The CP rank and Tucker rank are set at 10 and $[10, 10, 10]$, respectively. Experimental results from Fig. 6.14 indicate that the performance of ACP and ATD

---

[14]Video sequences: http://jacarini.dinf.usherbrooke.ca/
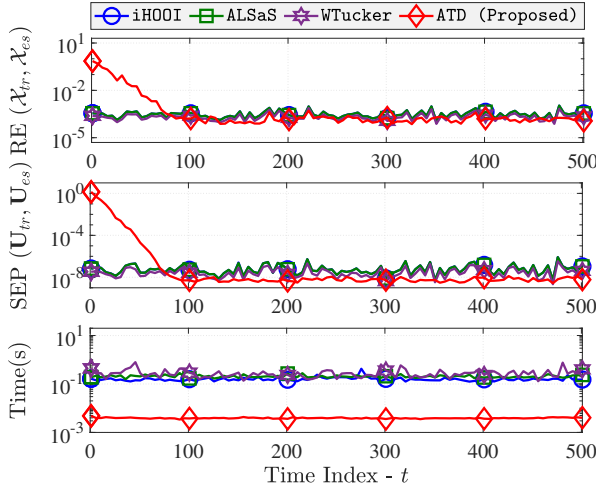
Figure 6.11: Performance of Tucker algorithms in the case where 50% **entries are observed and Tucker rank** $\mathbf{r_{TD}} = [3, 3, 3, 3]$**, and the noise level** $\sigma = 10^{-2}$**.**

is not much affected by the forgetting factor. For this task, we therefore keep the value of λ at 0.5 as in previous experiments on synthetic data.

We next compare our algorithms with OLSTEC [176], TeCPSGD [106] and CP-PETRELS [215]. We set the value of λ at 0.7, 0.001 and 0.999, respectively, for OLSTEC, TeCPSGD and CP-PETRELS. Besides, OLSTEC and TeCPSGD are also depended on the regularization parameter μ which value is fixed at 0.1. Performance of these algorithms is shown statistically in Tab. 7.1 and graphically in Fig. 6.15. We can see that ATD outperforms adaptive CP algorithms in almost all tests. ACP also provides reasonable estimation accuracy on these data as compared to others. CP-PETRELS seems to fail to track video background and thus recovers missing data unsuccessfully. With respect to the running time, experimental results indicate that ACP is the fastest adaptive tensor decompositions.

Multichannel EEG Analysis. We follow the experimental framework in [292, 346] to illustrate the use of ACP for analyzing multichannel EEG signals. In this task, we use a public EEG dataset collected on 14 subjects during the stimulation of hands[15]. The EEG signals are recorded using a system of 64 channels (electrodes) and we have 28 measurements per subject in total.

We construct three-order EEG tensor of measurement × channel × time − frequency by taking continuous wavelet transform to each EEG channel. Note that, the resulting time-frequency representations are reshaped into vectors of length 4392 and hence being streamed. In a nutshell, we have the EEG

---

[15]EEG data: http://www.erpwavelab.org/index.htm
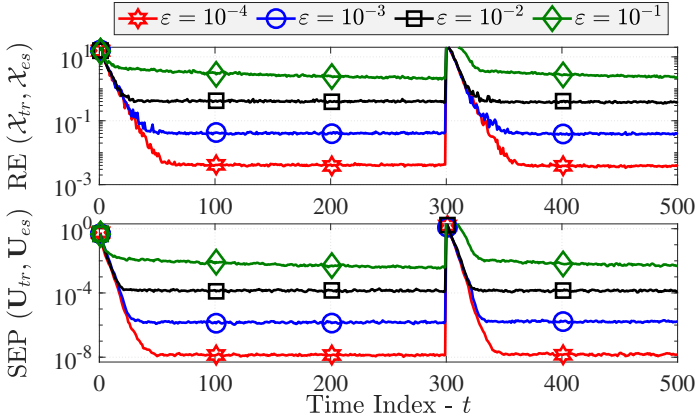
Figure 6.12: Effect of the time-varying factor ε on the performance of ATD: Tucker rank $[3, 3, 3, 3]$, 90% entries are observed, the noise level is $\sigma = 10^{-2}$ and an abrupt change at $t = 300$.

tensor whose size is $28 \times 64 \times 4392$ and its rank is set at 3 as provided in [292, 346]. At each time, data of 20 (and 40) channels are supposed to be discarded randomly for our missing observation purpose.

We evaluate the performance of ACP in a comparison with the adaptive NL-PETRELS algorithm in [292] and the batch CP-WOPT algorithm in [346]. To have a good initialization for NL-PETRELS, the 1500 first slices are used to construct the training tensor. Also, the forgetting factor λ is set at 0.999. By contrast, ACP is not as sensitive to initialization conditions, so it is initialized at random. We consider results obtained by using the batch algorithm as our ground truth.

Under the CP model with $r_{CP} = 3$, taking the tensor decomposition to the EEG tensor results in three loading factors $\mathbf{A} \in \mathbb{R}^{28 \times 3}$, $\mathbf{B} \in \mathbb{R}^{64 \times 3}$ and $\mathbf{C} \in \mathbb{R}^{4392 \times 3}$ corresponding to, respectively, the measurement, channel and time-frequency modes. Fig. 6.16 illustrates the estimation of $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ using CP-WOPT, NL-PETRELS and ACP. In particular, we use bar plots and 3D head plots to represent the column vectors of $\mathbf{A}$ and $\mathbf{B}$, while the time-frequency representations corresponding to the columns of $\mathbf{C}$ are plotted as matrices. We can see from Fig. 6.16 that both adaptive algorithms are capable of tracking three EEG loading factors. Indeed, our ACP provides a slightly better estimation as compared to that of CP-WOPT. However, in the presence of highly incomplete observations (e.g. 40 channels are missing), NL-PETRELS fails to estimate the EEG loading factors while our ACP algorithms still works well, as shown in Fig 6.17.
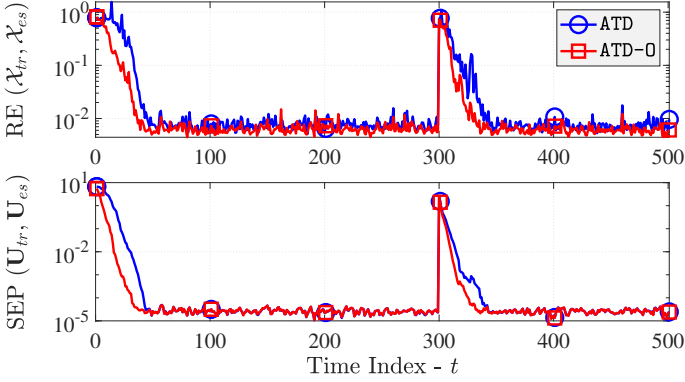
Figure 6.13: Comparison of ATD and ATD-O (orthogonality constraint) in a dynamic scenario: the time-varying factor $\varepsilon = 10^{-2}$, the noise level $\sigma = 10^{-3}$, 70% observations are observed and an abrupt change at $t = 300$.

6.4.3 Performance of RACP

We here provide several experiments on both synthetic and real data to demonstrate the effectiveness of RACP and its variant. In particular, the performance of our method is evaluated in comparison with the-state-of-the-art algorithms with respect to the following aspects: (i) impact of outliers, (ii) impact of missing data, and (iii) tracking ability in noisy and time-varying environments.

6.4.3.1 Experiment Setup

At $t = 0$, the loading factor $\mathbf{U}_0^{(n)} \in \mathbb{R}^{I_n \times r}$, $n = 1, 2, \ldots, N$ is randomly initialized whose entries are i.i.d. from a normal distribution $\mathcal{N}(0, 1)$. When $t \geq 1$, $\mathbf{U}_t^{(n)}$ is varied according to the following model:

$$\mathbf{U}_t^{(n)} = \mathbf{U}_{t-1}^{(n)} + \epsilon \mathbf{N}_t^{(n)}, \tag{6.108}$$

where $\mathbf{N}_t^{(n)}$ is a Gaussian noise matrix (with zero-mean and unit-variance), and ϵ is a positive time-varying factor used to control the variation of $\mathbf{U}^{(n)}$ between t and $t - 1$.

The t-th slice $\boldsymbol{\mathcal{Y}}_t$ is then generated under the data model

$$\boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{P}}_t \circledast \left(\left[\!\left[\{\mathbf{U}_t^{(n)}\}_{n=1}^{N-1}, \mathbf{u}_t^{(N)} \right]\!\right] + \boldsymbol{\mathcal{O}}_t + \boldsymbol{\mathcal{N}}_t \right), \tag{6.109}$$

where $\boldsymbol{\mathcal{P}}_t$ is a binary observation mask according to a Bernoulli distribution with probability of observing data $1 - \omega_{\text{miss}}$, $\boldsymbol{\mathcal{N}}_t$ is a Gaussian noise tensor

(a) ACP: Averaged Reconstruction Error

(b) ATD: Averaged Reconstruction Error

(c) ACP: full observations

(d) ATP: full observations

(e) ACP: 50% observations

(f) ATD: 50% observations
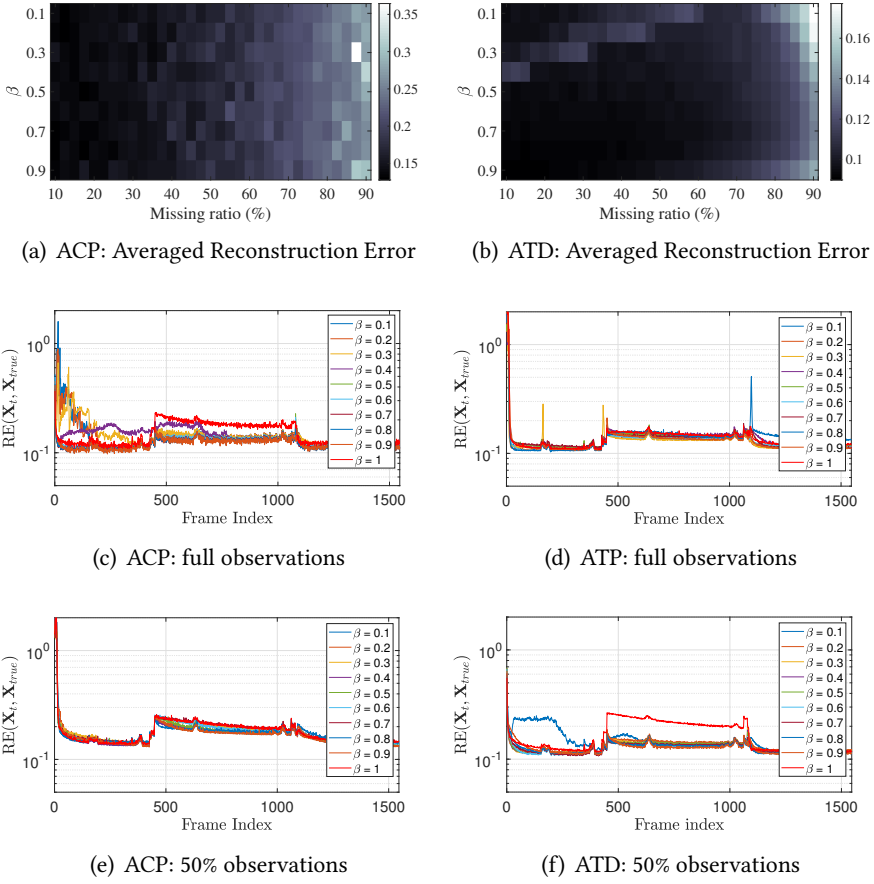
Figure 6.14: Effect of the forgetting factor β on the video completion accuracy of ACP and ATC on Lobby data.

with i.i.d. entries $\mathcal{N}(0, \sigma_n^2)$, $\boldsymbol{O}_t$ is a sparse outlier tensor whose entries are drawn uniformly from the range $[0, A_{\text{outlier}}]$ and the indices of outliers also follow a Bernoulli distribution with probability ω_{outlier}, and $\mathbf{u}_t \in \mathbb{R}^{r \times 1}$ is a standard normal random vector.

6.4.3.2 Robustness of RACP

We first investigate the robustness of RACP against gross data corruptions. Specifically, we change the density of outliers and missing data, and then measure the relative error between the ground truth and RACP's estimation.

In this task, we use a synthetic 4$^{\text{th}}$-order streaming tensor of size $20 \times 20 \times 20 \times 1000$ and the CP rank is set at $r = 2$ and $r = 5$. The noise level σ_n and the time-varying factor ϵ are fixed at 10^{-3} and 10^{-2}, respectively. We

(a) Lobby video: 50% missing.

(b) Lobby video: Performance of ATD.

(c) Hall video: 50% missing.

(d) Hall video: Performance of ATD.
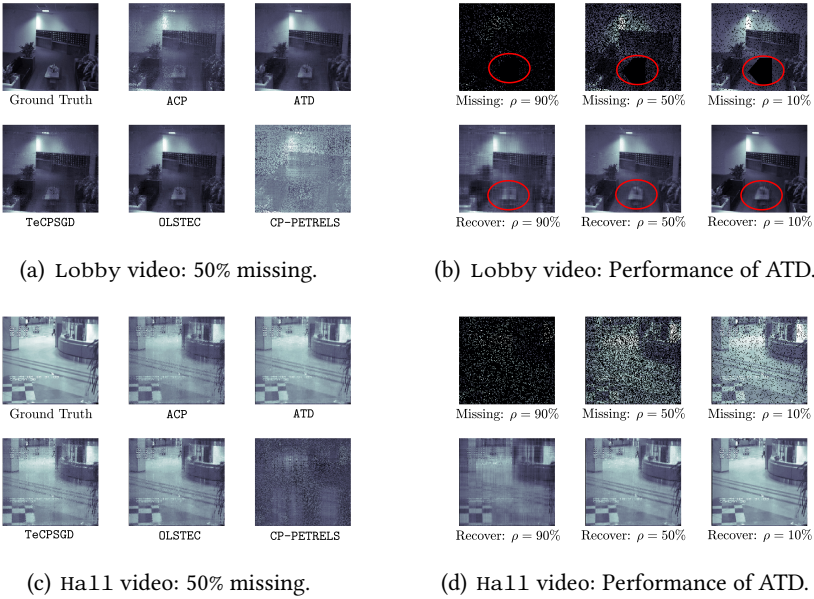
Figure 6.15: Performance of adaptive tensor completion algorithms on the video sequences.

(a) CP-OPT (Batch)   (b) NL-PETRELS (Adaptive)   (c) ACP (Adaptive)
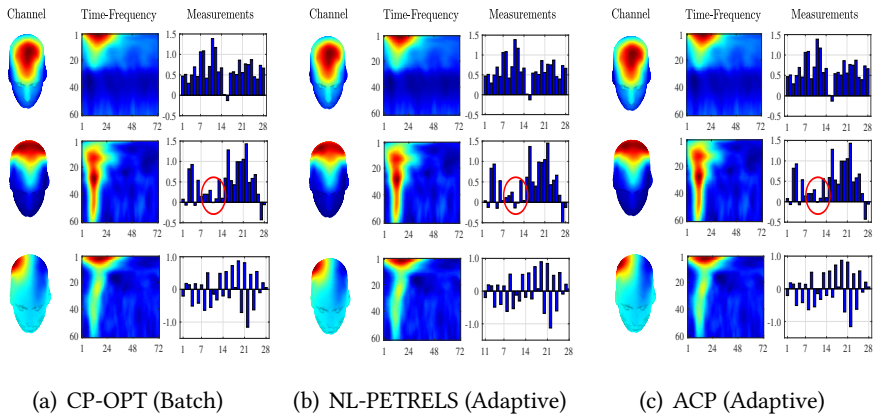
Figure 6.16: Waveform-preserving character of ACP on the EEG tensor: 20 **channels are missing.**

consider the case where the underlying data is corrupted by strong outliers with $A_{\text{outlier}} = 10$. The fraction of outliers (ω_{outlier}) and missing data (ω_{miss}) are varied in the range $[5\%, 95\%]$. Throughout our experiments, the forgetting factor λ is fixed at 0.5 while the window length is $L_t = t$.

Methods			TeCPSGD		OLSTEC		CP–PETRELS		ACP		ATD	
Dataset	Size	Missing	RE($\mathcal{X}$)	Time(s)	RE($\mathcal{X}$)	Time(s)	RE($\mathcal{X}$)	Time(s)	RE($\mathcal{X}$)	Time(s)	RE($\mathcal{X}$)	Time(s)
Highway $320 \times 240 \times 1700$		10%	0.2057	36.582	0.1693	132.02	0.9250	451.41	0.2178	14.437	0.1484	36.587
		50%	0.2111	35.252	0.1709	95.188	0.9346	273.98	0.2251	13.295	0.1526	33.269
		90%	0.2256	27.103	0.1849	54.246	0.9224	107.79	0.2725	13.017	0.1964	26.996
Hall $174 \times 144 \times 3584$		10%	0.1456	15.060	0.1247	83.789	0.9819	339.10	0.1457	11.852	0.1006	36.293
		50%	0.1450	14.916	0.1260	74.869	0.9269	188.15	0.1602	11.808	0.1045	31.576
		90%	0.1614	12.532	0.1497	47.235	0.9281	71.576	0.2341	11.897	0.1426	25.047
Lobby $128 \times 160 \times 1546$		10%	0.1324	5.672	0.1213	29.490	0.9161	107.44	0.1258	4.613	0.0868	14.590
		50%	0.1452	4.920	0.1228	21.940	0.8596	61.051	0.1881	4.711	0.0884	10.630
		90%	0.1733	4.022	0.1530	14.701	0.9736	22.150	0.2602	3.811	0.1333	9.245
Park $288 \times 352 \times 600$		10%	0.1057	10.303	0.0905	49.213	0.9945	186.28	0.1270	6.458	0.0686	16.157
		50%	0.1246	9.940	0.0916	33.660	0.9892	127.30	0.1441	5.825	0.0759	14.052
		90%	0.1369	8.497	0.1006	22.031	0.9627	50.435	0.2001	5.179	0.1122	10.966

Table 6.2: Performance of adaptive tensor decompositions on video data.

Phase transitions w.r.t. the pair of $\{\omega_{\texttt{outlier}}, \omega_{\texttt{miss}}\}$ are shown in Fig. 6.18. The results indicate that there is a large region in which our estimation was successful. Particularly, RACP worked well when the number of "clean" data is large enough. In the presence of huge data corruptions (e.g., $\omega_{\texttt{outlier}} \geq$ 70% and/or $\omega_{\texttt{miss}} \geq$ 70%), the proposed algorithm failed to track the underlying tensor model.

Next, we evaluate the tracking ability of RACP in time-varying environments. The two synthetic rank-5 tensors of size $20 \times 20 \times 20 \times 1000$ and $20 \times 20 \times 20 \times 20 \times 1000$ are used in this task. The fraction of missing entries and sparse outliers are both set to 5%. The outlier intensity $A_{\texttt{outlier}}$ and the noise factor σ_n are fixed at 10 and 10^{-4}, respectively. The value of the time-varying factor ϵ is varied from $[10^{-4}, 10^{-1}]$. An abrupt change is created at $t = 600$ to assess how fast RACP converges. We can see from Fig. 7.12

(a) CP-OPT (Batch)   (b) NL-PETRELS (Adaptive)   (c) ACP (Adaptive)
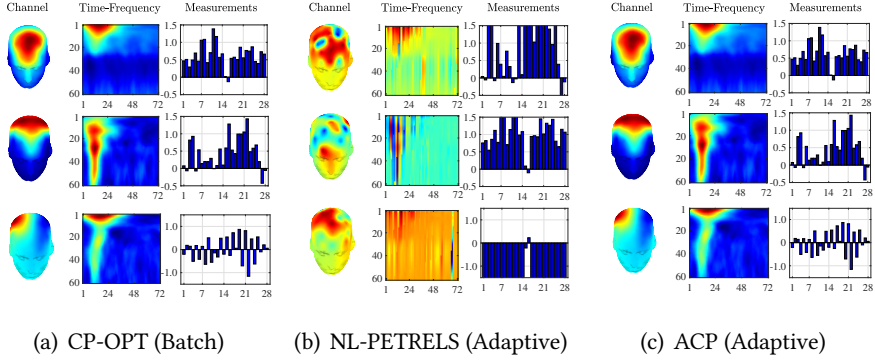
Figure 6.17: Waveform-preserving character of ACP on the EEG tensor: 40 channels are missing.

(a) $r = 2$ (b) $r = 5$
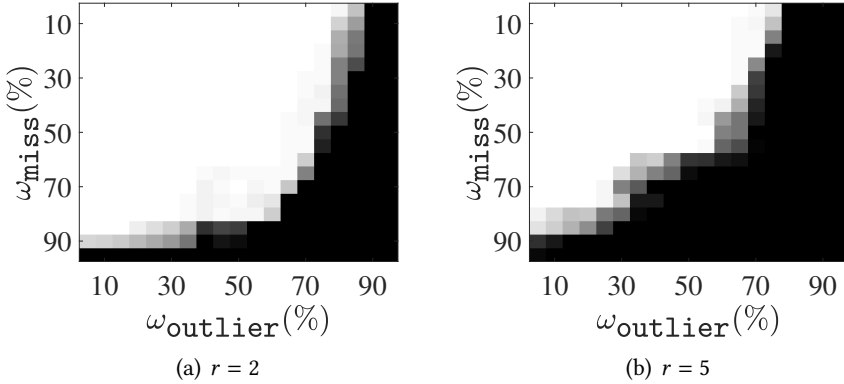
Figure 6.18: Effect of data corruptions (outliers and missing values) on performance of RACP. Black color denotes failure, white color denotes perfect estimation, and gray color is in between.

that RACP's convergence rate is not much affected by the value of ϵ but its estimation accuracy.

To demonstrate the effectiveness of the proposed algorithm, we compare the performance of RACP with the state-of-the-art adaptive CP decompositions, including TeCPSGD [106], OLSTEC [176], and ACP [30]. To have a fair comparison, algorithm parameters are set by default as suggested by their authors. These algorithms are dependent on a forgetting factor; we set its value at 0.7, 0.001, and 0.5 for OLSTEC, TeCPSGD, and ACP, respectively. The penalty parameter is set at 10^{-3} and 10^{-1} for OLSTEC and TeCPSGD, respectively.

Since OLSTEC and TeCPSGD are only capable of tracking third-order

(a) 4th-order: $20 \times 20 \times 20 \times 1000$ (b) 5th-order: $20 \times 20 \times 20 \times 20 \times 1000$
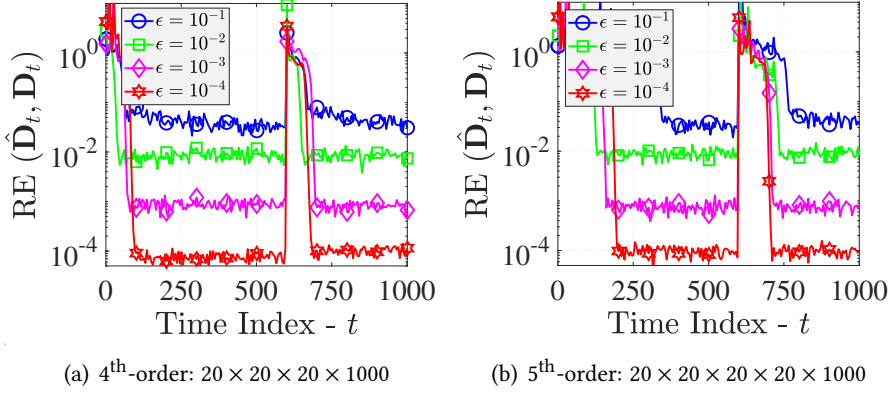
Figure 6.19: Performance of RACP in time-varying environments.

(a) $A_{\text{outlier}} = 1$ (small)   (b) $A_{\text{outlier}} = 10$ (strong)
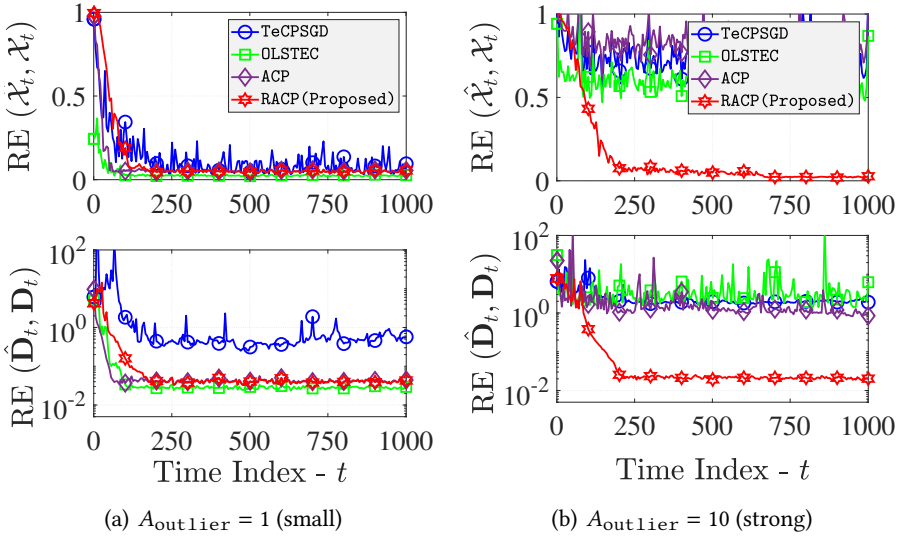
Figure 6.20: Impact of outlier intensity (A_{outlier}) on performance of adaptive CP algorithms; $\omega_{\text{miss}} = 10\%$, $\omega_{\text{outlier}} = 20\%$, $\sigma = 10^{-2}$, $\varepsilon = 10^{-2}$.

streaming tensors, we here use a synthetic streaming tensor of size $20 \times 20 \times 1000$ and its rank is fixed at 5. The noise level and time-varying factor are both kept at 10^{-2}. Performance comparison results are shown in Figs. 6.20 and 6.21.

Fig. 6.20 illustrates the impact of the outlier intensity on the performance of the four adaptive CP algorithms in the presence of 10% missing data and 20% outliers. When the outlier intensity is small, all algorithms could track the underlying tensor model over time, as shown in Fig. 6.20(a). Indeed, TeCPSGD
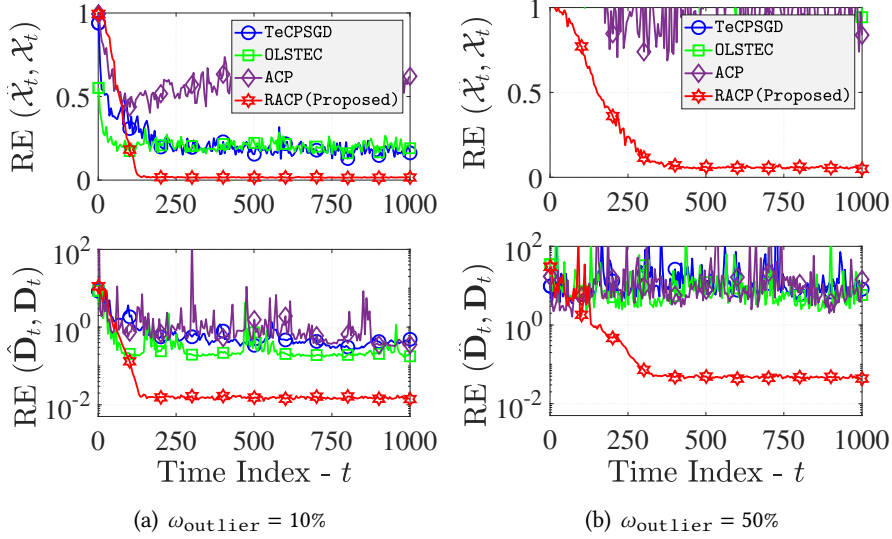
Figure 6.21: Impact of outlier density ($\omega_{\mathtt{outlier}}$) on performance of adaptive CP algorithms: $\omega_{\mathtt{miss}} = 10\%$, $\sigma = 10^{-2}$, $\varepsilon = 10^{-2}$, $A_{\mathtt{outlier}} = 10$.

yielded a worse estimation than the three remaining adaptive CP algorithms. In the presence of strong outliers, the state-of-the-art adaptive CP algorithms failed to update the tensor basis and recover the corrupted tensor slice. By contrast, our RACP algorithm still worked well, as shown in Fig. 6.20(b). Fig. 6.21 illustrates the impact of the outlier density on the performance of RACP against the three adaptive CP algorithms when the missing density $\omega_{\mathrm{miss}} = 10\%$ and outlier intensity $A_{\mathrm{outlier}} = 10$. We can see that RACP outperformed OLSTEC, TeCPSGD, and ACP in all testing cases. Similar to the case study of strong outliers, the state-of-the-art adaptive algorithms were unable to track the streaming tensors when the number of outliers is large.

We next investigate the performance of RACP when the loading factors are not normal in comparison with other adaptive CP algorithms. In particular, the initial factors $\{\mathbf{U}_0^{(n)}\}_{n=1}^N$ are sampled from a uniform distribution on the $(0, 1)$ interval instead of Gaussian one. The time-varying model (6.108) is replaced with $\mathbf{U}_t^{(n)} = \mathbf{U}_{t-1}^{(n)} + \epsilon \mathbf{N}_t^{(n)}$ where $\mathbf{N}_t^{(n)}$ is also an i.i.d. uniform random matrix from 0 to 1. The parameter specifications are kept as in the previous experiment. Results are illustrated in Fig. 6.22. We can see that the proposed RACP algorithm still tracks successfully the loading factors along the time while the state-of-the-art CP algorithms failed.

Experimental results in Figs. 6.20, 6.21, and 6.22 suggest that the outlier rejection step (e.g. Step 1 in RACP) using the ADMM solver plays an important role in the tracking process when observations are corrupted by sparse

(a) $\omega_{\texttt{miss}} = \omega_{\texttt{outlier}} = 5\%$ (b) $\omega_{\texttt{miss}} = \omega_{\texttt{outlier}} = 20\%$
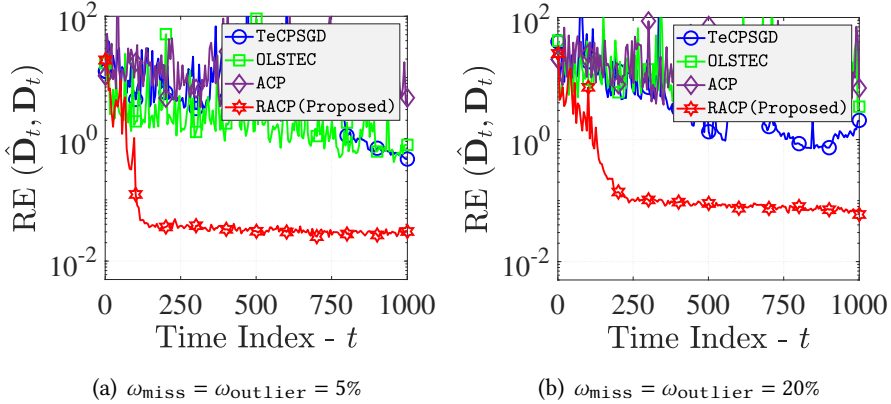
Figure 6.22: Non-Gaussian loading factors.

outliers. Therefore, we next evaluate the effectiveness of the proposed outlier rejection by applying the ADMM solver to other trackers: TeCPSGD and OL-STEC. We here reuse the experiment setup above and create an abrupt change at $t = 600$. We can see from Fig. 6.23 that the combination of the ADMM solver and OLSTEC resulted in the best convergence rate and estimation accuracy. This is probably due to the effectiveness of the second-order estimator in slowly time-varying environments. Our RACP provided a reasonable performance compared to that of OLSTEC, while TeCPSGD tracker did not work well. It should note that OLSTEC is designed for only 3rd-order streaming tensors and its computational complexity is high indeed. Our tracker is much faster and capable of dealing with higher-order streaming tensors. We refer the readers to our companion work in [30] for further comparisons of ACP against TeSGD and OLSTEC.

Finally, we conduct a performance comparison between the original RACP and its variant in which the step of re-updating $\boldsymbol{\mathcal{P}}_t$ defined as in (6.61) is used. We reuse the two rank-5 tensors of size $20 \times 20 \times 20 \times 1000$ and $20 \times 20 \times 20 \times 20 \times 1000$. The fraction of missing entries is fixed at 10%. We set the outlier density and intensity to 10% and 10, respectively. The noise and time-varying factors are kept at 10^{-2} and an abrupt change at $t = 600$ is also created as in previous experiments. The results are illustrated in Fig. 6.24. As can be seen the outlier rejection mechanism can help improve the convergence rate of RACP.

(a) $\omega_{\mathtt{miss}} = \omega_{\mathtt{outlier}} = 5\%$ (b) $\omega_{\mathtt{miss}} = \omega_{\mathtt{outlier}} = 20\%$
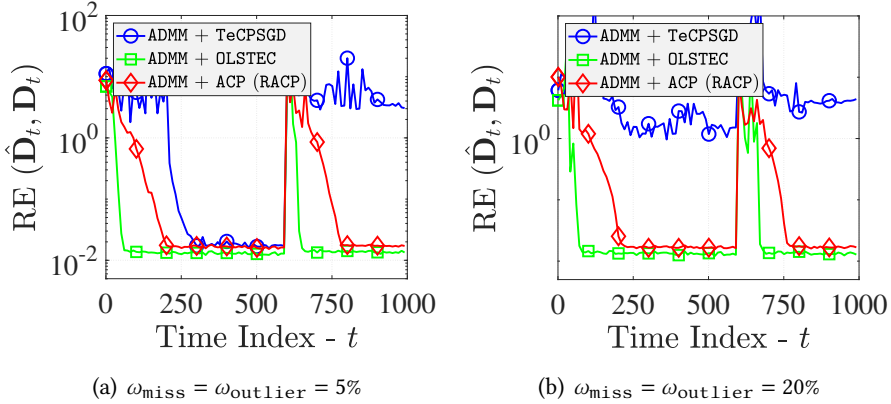
Figure 6.23: Outlier rejection with different trackers.

6.4.3.3 Nonnegative RACP

We reuse the experiment setup in Section 6.4.3.1, but the time variation of $\mathbf{U}^{(n)} \geq 0$ is modified as

$$\mathbf{U}_t^{(n)} = \mathrm{abs}\left(\mathbf{U}_{t-1}^{(n)} + \epsilon \mathbf{N}_t^{(n)}\right), \tag{6.110}$$

where $\mathrm{abs}(\cdot)$ denotes the absolute value, $\mathbf{N}_t^{(n)}$ is a Gaussian noise matrix with i.i.d. entries, and ϵ is to control the variation.

We first investigate the performance of NRACP against time-varying environments. A synthetic rank-5 nonnegative tensor of size $50 \times 50 \times 50 \times 1000$ is used in this task. We consider the case where 10% of the measurements are corrupted by outliers with $A_{\mathtt{outlier}} = 10$ and the noise level is $\sigma_n = 10^{-3}$. An abrupt change at $t = 600$ is created to evaluate how fast NRACP converges. The results are shown in Fig. 6.25. We can see that the relative error between the estimation and ground truth converged to an error floor. Furthermore, the missing density $\omega_{\mathtt{miss}}$ impacted only the convergence rate of NRACP. Specifically, the lower the missing density $\omega_{\mathtt{miss}}$ was, the faster NRACP converged.

Next, we study the robustness of NRACP against the noise variance in comparison with NSOAP [174] and NsTEF [347]. Since both two algorithms are only feasible for third-order tensors without corruptions (outliers and missing values), we use a synthetic outlier-free tensor of size $50 \times 50 \times 1000$ and rank 5 for this task. The time-varying factor ϵ is set at 10^{-3}. Performance comparison results are illustrated in Fig. 6.26. At a low SNR, NSOAP provided a better estimation accuracy than NRACP and NsTEF. However, the proposed NRACP outperformed NSOAP and NsTEF at the high SNR, see Fig. 6.26(b). In the presence of abrupt changes, the convergence rate of NRACP was fast while NSOAP and NsTEF failed to track the change.

(a) 4th-order: $20 \times 20 \times 20 \times 1000$ (b) 5th-order: $20 \times 20 \times 20 \times 20 \times 1000$
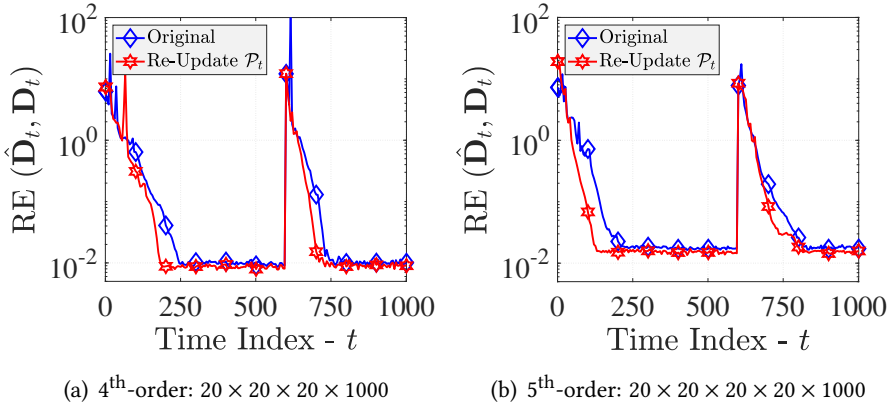
Figure 6.24: Convergence rate of RACP and its modification with the re-update of $\mathcal{P}_t$ as defined in (6.61)**:** $\omega_{\texttt{miss}} = 10\%$, $\omega_{\texttt{outlier}} = 10\%$, $A_{\texttt{outlier}} = 10$, $\sigma = 10^{-2}$**, and** $\varepsilon = 10^{-2}$**.**
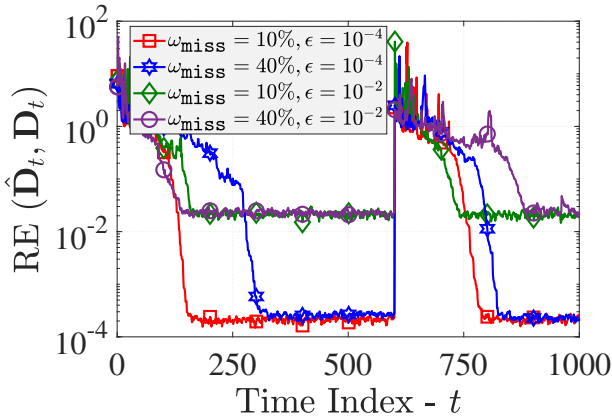
Figure 6.25: Incomplete observations & time-varying scenarios: Performance of NRACP on a synthetic rank-5 tensor of size $50 \times 50 \times 50 \times 500$**;** $\sigma_n = 10^{-3}$**,** $A_{\texttt{outlier}} = 10$**,** $\omega_{\texttt{outlier}} = 10\%$**.**

6.4.3.4 Real Datasets

To demonstrate the use of RACP with real-world datasets, we consider the following tasks: (i) tracking the online low-rank approximation of real-world data streams, (ii) multichannel EEG analysis, and (iii) video background modeling and foreground detection. Please see Tab. 6.3 for a summary of real datasets used in this paper.

(a) $\sigma_n = 10^{-1}, \epsilon = 10^{-3}$ (b) $\sigma_n = 10^{-3}, \epsilon = 10^{-3}$
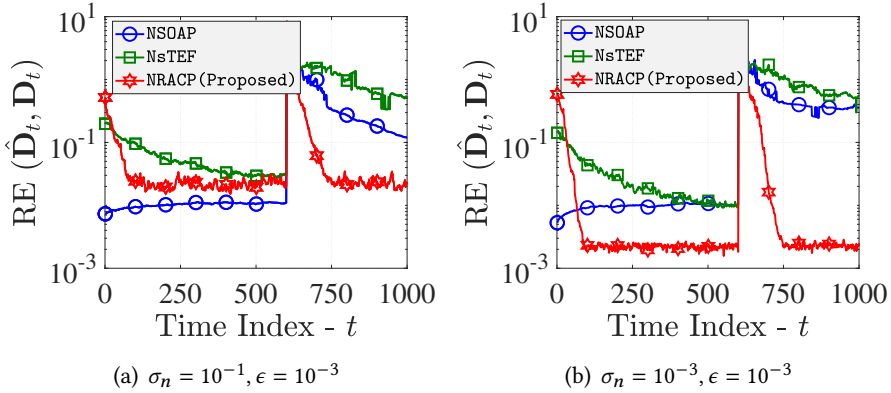
Figure 6.26: Nonnegative adaptive CP decompositions: Outliers-free, full observations and an abrupt change at $t = 600$**.**

Table 6.3: Real datasets under the study.

Dataset		Data size	Tasks
Intel Berkeley Lab		$54 \times 4 \times 1152$	Tracking the online low-rank approximation & online data completion
Internet Traffic		$12 \times 12 \times 48384$	
Taxi Trip Record		$265 \times 265 \times 3672$	
Video	Hall	$176 \times 144 \times 3584$	Background modeling & foreground detection
	Lobby	$128 \times 160 \times 1546$	
	Highway	$240 \times 320 \times 1700$	
EEG	ERPWAVELAB	$28 \times 64 \times 4392$	Multichannel EEG analysis & anomaly EEG detection
	Epileptic data	$19 \times 500 \times 6929$	

Task 1: Tracking the online low-rank approximation and online data completion

Datasets: In this task, we use three real datasets: Intel Berkeley Lab[16], Internet Traffic[17], and Taxi Trip Record[18]. The first dataset is a collection of timestamped topology information gathered from 54 positions (sensors) in the Intel Berkeley Research Lab. Specifically, these sensors collected: temperature (in degree Celsius), humidity (ranging from 0% to 100%), light (in Lux), and voltage (in volt, ranging from 2 to 3). Accordingly, we represent the sensor data by a three-order tensor of size $54 \times 4 \times 1152$ (i.e., *sensor* $\times$ *measurement* $\times$ *time*). The

---

[16]Intel Berkeley Lab: http://db.csail.mit.edu/labdata/labdata.html

[17]Internet Traffic: https://roughan.info/project/traffic_matrix/

[18]Taxi Record: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
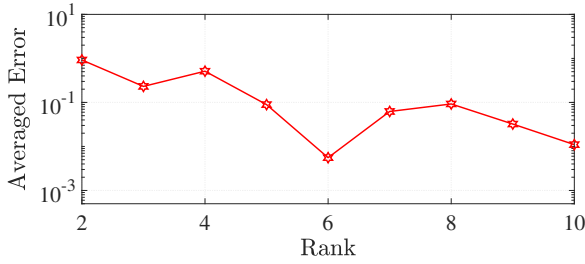
(a) Performance of RACP with different values of tensor rank
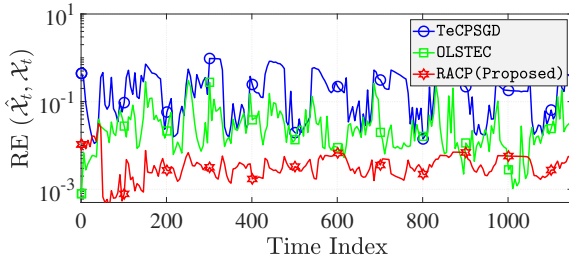
(b) Performance of adaptive CP algorithms with tensor rank $r = 6$

Figure 6.27: Experimental results on the Intel Berkeley Lab data.

second dataset is the link traffic data which was collected from the Internet2 backbone network Abilene. The Abilene backbone is relatively small with 12 routers, 15 links, and 144 flow entries in each traffic matrix of size 12×12. We concatenate all these traffic matrices into a tensor of size $12 \times 12 \times 48384$. The third dataset describes yellow taxis trip records in the pairs of 265 pick-up and drop-off sites in New York. Each trip record contains several attributes, such as pick-up/drop-off times and locations, elapsed trip distance, rate type, and payment method. In this work, we specifically construct a third-order tensor of size $265 \times 265 \times 3672$ (i.e., *origin* $\times$ *destination* $\times$ *time*).

Experiments & Results: Following the same experiment setup in subsection 6.4.3.1, data corruptions are generated as follows. The locations of missing entries and sparse outliers are randomly generated with probabilities $\omega_{\texttt{miss}}$ and $\omega_{\texttt{outlier}}$, respectively. Outlier's values are drawn uniformly from the range $[0, \max(\boldsymbol{X})]$ where $\max(\boldsymbol{X})$ is the largest absolute value in the underlying data $\boldsymbol{X}$. In this experiment, we choose the value of $\omega_{\texttt{miss}}$ and $\omega_{\texttt{outlier}}$ among the range $\{5\%, 10\%, 20\%, 40\%\}$. As the true rank is unknown, we first vary its value from 2 to 10 and then choose the "best" one based on the averaged reconstruction error, see Fig. 6.27(a) for an example. We compare the performance of RACP against the two adaptive CP algorithms TeCPSGD [106] and OLSTEC [176]. Both algorithms are dependent on the forgetting factor
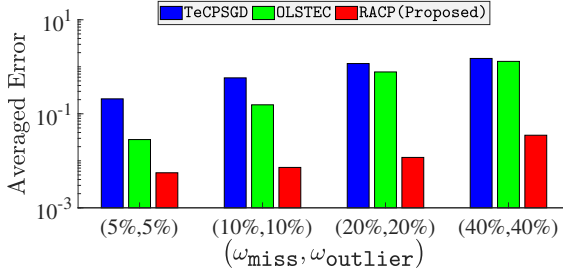
(a) Intel Berkeley Lab: Estimated rank $r = 6$
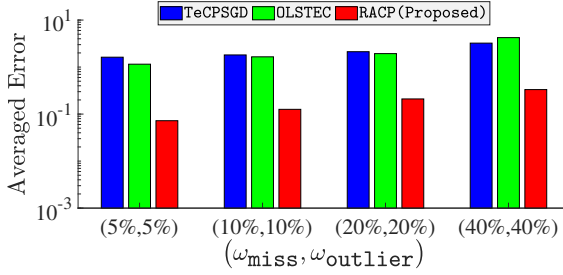
(b) Internet Traffic: Estimated rank $r = 3$
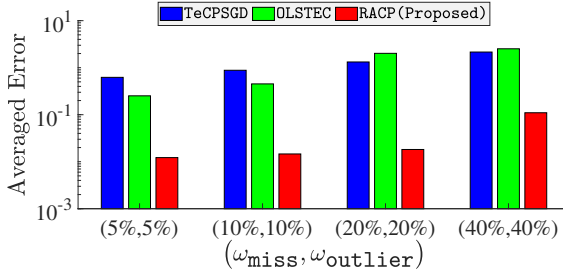
(c) Taxi Trip Record: Estimated rank $r = 8$

Figure 6.28: Completion accuracy of adaptive CP algorithms on real-world data streams.

λ, and its value is set at 0.98, 0.001, and 0.7, respectively. The penalty parameter μ is set at 1 for both TeCPSGD and OLSTEC. The experimental result in Fig. 6.28 indicates that RACP outperforms TeCPSGD and OLSTEC.

Task 2: Multichannel EEG Analysis

<u>Datasets</u>: In this task, we use two public EEG datasets: ERPWAVELAB[19] and Epileptic EEG Data[20]. The former dataset contains wavelet transformed ver-

---

[19]ERPWAVELAB: http://www.erpwavelab.org/
[20]Epileptic EEG Data: https://data.mendeley.com/datasets/5pc2j46cbc/1

(a) Normal data   (b) Complex partial seizure
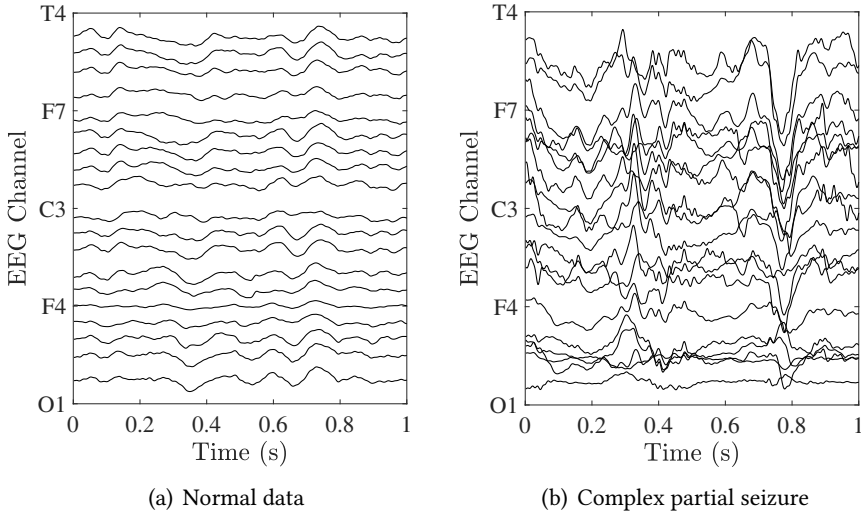
Figure 6.29: Epileptic EEG Dataset.

sions of EEG signals that were collected from 14 subjects during the hand
stimulation (i.e., proprioceptive pulls of the left and right hands) for inter-
trial phase coherence analysis. In particular, these EEG signals were recorded
using an electrode system of 64 channels with 28 measurements per subject.
The continuous wavelet transform was then applied to represent these signals
in the time-frequency domain. The latter dataset includes 20 EEG recordings
of 6 patients diagnosed with epilepsy at the American university of Beirut
medical center. The EEG data were particularly recorded by using a system
of 21 channels with the sampling rate of 500Hz. The dataset includes 3895
normal segments and 3850 abnormal segments in which there are 3034 par-
tial seizures, 705 electrographic seizures, and 111 video-detected seizures with
no visual change over EEG. Figs. 6.29(a) and 6.29(b) illustrate EEG normal
waveforms and complex partial seizures. In what follows, we consider two
common problems in multichannel EEG analysis: (i) incomplete multichannel
EEG analysis from partial observations and (ii) anomaly EEG detection.

Incomplete Multichannel EEG Analysis: Here, we use the ERPWAVELAB
dataset and follow the same experimental setup in [30, 292, 346] to demon-
strate the use of RACP with real EEG signals. Particularly, we construct an
EEG tensor of size $28 \times 64 \times 4392$ (i.e., *measurement × channel × time-frequency*).
To generate incomplete observations, signals from some channels at each
time are supposed to be missing at random. As suggested in [292, 346], we set
the tensor rank at $r = 3$. Performance of RACP is compared with two adaptive
CP algorithms NL-PETRELS [292] and ACP [30]. We fix the forgetting factor

Table 6.4: Averaged errors of adaptive CP algorithms for multichannel EEG analysis from incomplete observations.

Missing channels	NL-PETRELS	ACP	RACP (Proposed)
1/64	0.051	0.063	0.056
10/64	0.062	0.025	0.023
20/64	0.077	0.011	0.014
30/64	0.121	0.097	0.086
40/64	0.891	0.132	0.119
50/64	1.325	1.137	0.982

λ at 0.999 and 0.5 for NL-PETRELS and ACP, respectively. As NL-PETRELS requires a warm start, we run the batch CP-WOPT algorithm [346] with the first 1500 tensor slices. Meanwhile, we use random initialization for ACP and RACP. In this experiment, we aim to factorize the EEG tensor into three basis components w.r.t. spatial domain, time-frequency domain, and measurement mode. As there is no real ground truth, we use the results (i.e., CP factors) derived from applying the batch CP-ALS algorithm to the EEG tensor with full observations as benchmarks. Experimental results are shown in Tab. 6.4 and Fig. 6.30. They indicate that RACP outperforms NL-PETRELS and provides a slightly better estimation than ACP, especially in the presence of highly incomplete observations (e.g., ≥ 40 channels are missing).

Anomaly EEG Detection: We demonstrate the use of RACP to detect abnormal activities in the brain (i.e., epileptic seizures) with the epileptic EEG dataset. Here, we adopt a simple but effective way to predict abnormalities in multidimensional data streams [237]. In particular, we model the abnormality of a tensor (streaming) slice $\boldsymbol{\mathcal{Y}}_t$ by its recovery error

$$e_t = \left\| \boldsymbol{\mathcal{P}}_t \circledast \left(\boldsymbol{\mathcal{Y}}_t - \boldsymbol{\mathcal{Y}}_t \prod_{n=1}^{N} \times_n \mathbf{U}_t^{(n)} \mathbf{U}_t^{(n)\#} \right) \right\|_F \Big/ \left\| \boldsymbol{\mathcal{Y}}_t \right\|_F, \qquad (6.111)$$

where $\{\mathbf{U}_t^{(n)}\}_{n=1}^{N}$ is the set of solutions generated by RACP at time t. It is also worth noting that the error e_t is relatively proportional to the norm of the outlier $\boldsymbol{O}_t$. We label $\boldsymbol{\mathcal{Y}}_t$ based on the following rule

$$e_t \underset{\text{normal}}{\overset{\text{abnormal}}{\gtrless}} \tau_t = \text{mean}\left(\{e\}_{L_t} \right) + \alpha \, \text{std}\left(\{e\}_{L_t} \right), \qquad (6.112)$$

where $\{e\}_{L_t}$ denotes the set of e_τ with $t - L_t < \tau \leq t$.

(a) Ground Truth

(b) ACP

(c) NL-PETRELS

(d) RACP (Proposed)
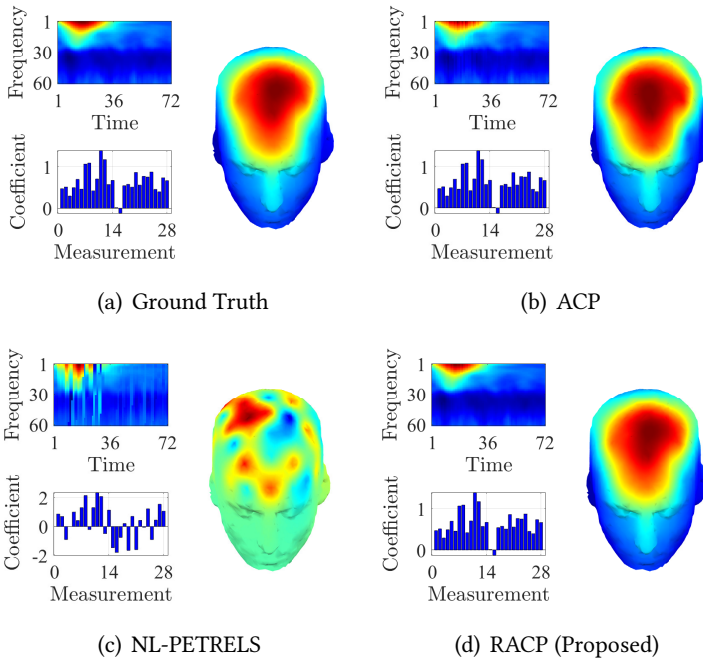
Figure 6.30: First component of EEG factors when 40/60 EEG channels are missing.

We follow the method in our companion work on epileptic spike detection [179] to obtain the time-frequency representation of multichannel EEG segments (including normal data and seizures), and hence the corresponding EEG tensors of size $19 \times 20 \times 500$ (i.e., *channel* $\times$ *scale* $\times$ *time*).[21] The resulting tensors are then concatenated into a huge tensor whose the last mode is being streamed. We use the first 100 tensors of normal data to obtain a warm start and the estimated rank of 9. Experimental results are shown in Fig. 6.31 (the error e_t over time) and Tab. 6.5 (prediction accuracy versus the value of α). Although the results are not really excellent, it is highly potential to detect anomalies in EEG signals by monitoring the approximation error. Subsequent investigations (e.g., type of wavelet, dominant scales, and mother function) are necessary to obtain a better prediction.

---

[21]As indicated in the EEG dataset description report, data of two channels Cz and Pz were omitted. Thus, we have 19 EEG channels left and each channel contains 500 samples. Also, 20 wavelet scales are chosen in the range [4, 8].
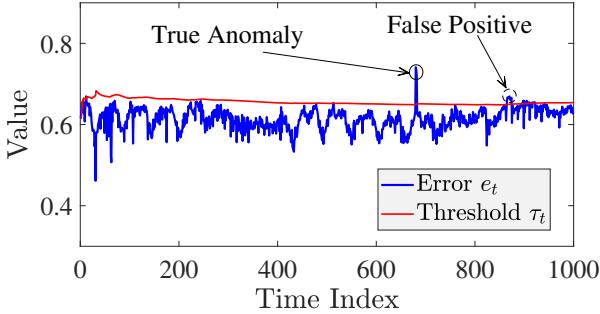
Figure 6.31: The error e_t over time with $\alpha = 1.5$ and $L_t = t$. Normal data which are inaccurately labelled as abnormal are referred to as "false positive".

Table 6.5: Anomaly EEG detection results. Sensitivity and specificity measure the percentage of anomaly and normal data detected correctly, respectively. Accuracy indicates the overall.

Value of α	Sensitivity	Specificity	Accuracy
0.1	42.21%	53.02%	47.57%
0.5	59.74%	66.48%	63.09%
1	72.80%	74.38%	73.59%
1.5	81.58%	85.16%	83.36%
2	50.16%	53.54%	51.83%

Task 3: Video Background-Foreground Modeling

To demonstrate the use of RACP for real applications, we consider the problem of video background modeling and foreground detection. Three real video sequences are used in this task, including Hall, Lobby, and Highway[22] (see Fig. 6.32). In particular, the Hall video is a set of 3584 images taken at an airport hall, and the image resolution is 176×144. The Lobby video contains 1546 images of size 128×160 pixels which was captured in an indoor office with switching on/off lights. The Highway video contains 1700 images of vehicles on a highway, and each frame is of size 240×320 pixels.

Background Modeling. We first measure the video background modeling ability of RACP in comparison with a robust subspace tracking algorithm PETRELS-ADMM [25], and two adaptive CP algorithms (TeCPSGD [106] and OLSTEC [176]). These algorithms are dependent on the forgetting factor λ,

---

[22]CDNET: http://jacarini.dinf.usherbrooke.ca.

a) Hall   b) Lobby   c) Highway

Figure 6.32: Three video surveillance sequences.

Original Frame   Corrupted Frame   TeCPSGD   OLSTEC   PETRELS-ADMM   RACP
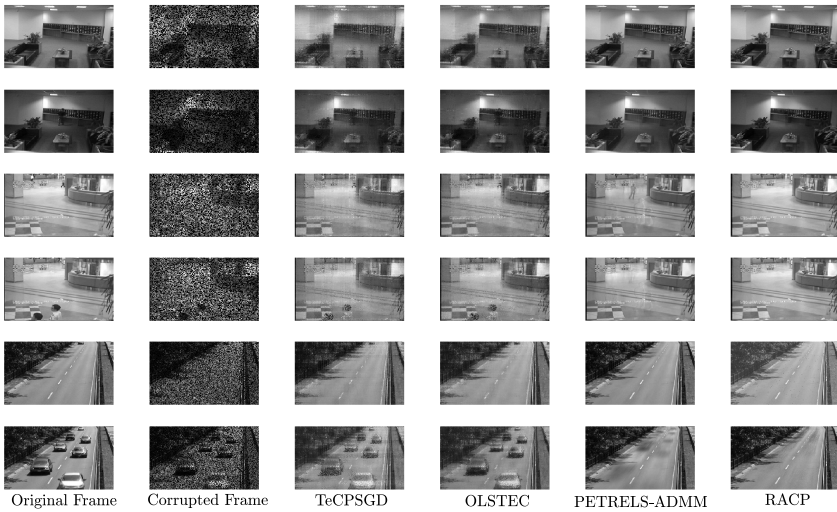
Figure 6.33: Qualitative illustration of video background modeling results.

and its value is set at 0.98, 0.001, and 0.7, respectively. The penalty parameter μ is set at 0.1 for both TeCPSGD and OLSTEC. The CP rank and subspace rank are set at 10.

We consider the scenario where 50% of pixels are supposed to be missing at random. Experimental results are illustrated in Fig. 6.33. As we can see that the two robust algorithms PETRELS-ADMM and RACP were able to recover the video background. Particularly, the proposed RACP provided slightly better estimation than PETRELS-ADMM. The two adaptive CP algorithms TeCPSGD and OLSTEC seem to have failed when the video frame contains moving objects, probably because they do not account for sparse outliers.

Foreground Detection. Next, we investigate the ability of RACP in

| | Original Frame | OSTD | GRASTA | PETRELS-ADMM | RACP |

Figure 6.34: Qualitative illustration of video foreground detection results.

video foreground detection. We also compare the performance of RACP with 3 notable foreground detection algorithms, including GRASTA [50], OSTD [278] and PETRELS-ADMM [25]. To have a fair comparison, algorithm parameters are set by default as suggested by their authors. Particularly, the penalty parameter ρ and constant step-size scale C are, respectively, set at 1.8 and 2 in GRASTA. The forgetting factor in PETRELS-ADMM is fixed at $\lambda = 0.98$, while OSTD is a parameter-free algorithm. As can be seen from Fig. 6.34 that RACP was capable of detecting moving objects in video streams and provided a competitive performance as compared to GRASTA, OSTD, and PETRELS-ADMM.

6.5   Conclusions

In this chapter, we have proposed three new low-complexity algorithms (including ACP, ATD, and RACP) for adaptive decomposition of higher-order incomplete and streaming tensors. First, developed based on CP decomposition, ACP estimates a multilinear LRA of streaming tensors from noisy and high-

dimensional data with high accuracy, even when the decomposition model may change slowly with time. Second, developed based on Tucker decomposition, ATD is a fast randomized tracker, able to recover missing entries from highly incomplete observations. Leveraging the stochastic approximation and the uniform sampling technique, ATD has been shown to be one of the fastest Tucker algorithms, much faster than the batch algorithms while providing good estimation accuracy. Third, a novel robust adaptive CP decomposition called RACP has been proposed to track the low-rank approximation of streaming tensors from uncertain, noisy, and imperfect measurements. Its convergence analysis has been established to guarantee that the solution generated by RACP converges to a stationary point asymptotically. Experimental results have indicated that all three algorithms could estimate the tensor factors as well as track their variations over time with high accuracy, and that they outperformed the state-of-the-art tensor tracking algorithms in both simulated and real data tests.

6.6 Appendix

6.6.1 Appendix A: Proof of Lemma 9

Our analysis follows the same framework to derive the asymptotic convergence of adaptive algorithms for problems of online matrix and tensor factorization [25, 106, 120, 121]. In particular, the convergence analysis contains three main stages: (I) we show that the solutions $\{\mathcal{U}_t, \mathbf{u}_t\}_{t=1}^{\infty}$ are uniformly bounded to justify the well-definedness condition. Their variations between two successive time instances satisfy $\|\mathbf{U}_{t+1}^{(n)} - \mathbf{U}_t^{(n)}\|_F \rightarrow \mathcal{O}(1/t)$ a.s. (II) The sequence of nonnegative surrogate values $\{g_t(\mathcal{U}_t)\}_{t=1}^{\infty}$ is quasi-martingale and convergent almost surely. (III) The empirical loss function $\{f_t(\mathcal{U}_t)\}_{t=1}^{\infty}$ and its surrogate $\{g_t(\mathcal{U}_t)\}_{t=1}^{\infty}$ converge to the same limit, i.e., $g_t(\mathcal{U}_t) \rightarrow f_t(\mathcal{U}_t)$ a.s. Accordingly, $\{\mathcal{U}_t\}_{t=1}^{\infty}$ converges to a stationary point of $f_t(\mathcal{U})$, i.e. $\nabla f_t(\mathcal{U}_t) \xrightarrow{t\to\infty} 0$.

6.6.1.1 Stage I

In order to justify the well-definedness condition, we first indicate that solutions $\{\mathcal{U}_t, \mathbf{u}_t^{(N)}\}_{t=1}^{\infty}$ are bounded and hence obtain several important propositions for the next stages[23].

---

[23]Note that we assume that the underlying tensor slices and their true loading factors are bounded, while in this analysis, we investigate the bound of solutions generated by the proposed ACP algorithm.

Proposition 12 *Solutions $\{\boldsymbol{\mathcal{U}}_t, \mathbf{u}_t\}_{t=1}^{\infty}$ generated by ACP are bounded.*

Proof. We first note that ACP begins with full column rank and bounded factors $\{\mathbf{U}_0^{(n)}\}_{n=1}^{N}$. The matrix $\mathbf{S}_0^{(n)}$ is initialized by a scaled identity matrix $\mathbf{S}_0^{(n)} = \delta_n \mathbf{I}_r$ with $\delta_n > 0$.

At each time $t > 0$, the coefficient vector $\mathbf{u}_t^{(N)}$ is achieved by minimizing the regularized LS problem

$$\mathbf{u}_t^{(N)} = \underset{\mathbf{u} \in \mathbb{R}^r}{\operatorname{argmin}} \left\| \mathcal{L}(\mathbf{y}_{\Omega_t} - \mathbf{H}_{\Omega_t} \mathbf{u}) \right\|_2^2 + \frac{\alpha}{2} \|\mathbf{u}\|_2^2. \tag{6.113}$$

At $\mathbf{u} = \mathbf{0}$, we obtain $\left\| \mathcal{L}(\mathbf{y}_{\Omega_t}) \right\|_2^2 \geq \left\| \mathcal{L}(\mathbf{y}_{\Omega_t} - \mathbf{H}_{\Omega_t} \mathbf{u}_t) \right\|_2^2 + \frac{\alpha}{2} \|\mathbf{u}_t\|_2^2$ and hence

$$\|\mathbf{u}_t^{(N)}\|_2^2 \leq \frac{2}{\alpha} \left\| \mathcal{L}(\mathbf{y}_{\Omega_t}) \right\|_2^2 < +\infty, \tag{6.114}$$

thanks to the assumption (A-1) that observed slides $\{\boldsymbol{\mathcal{Y}}_t\}_{t \geq 1}$ are bounded. It implies that the solution $\mathbf{u}_t$ is bound.

In the following steps, we use the mathematical induction to indicate the bound of $\boldsymbol{\mathcal{U}}_t$.

The base case. We prove that the set of solutions $\boldsymbol{\mathcal{U}}_1 = \{\mathbf{U}_1^{(n)}\}_{n=1}^{N-1}$ is bounded at $t = 1$.
Recall that, the minimizer $\mathbf{U}_1^{(n)}$ is derived from the following optimization

$$\mathbf{U}_1^{(n)} = \underset{\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times r}}{\operatorname{argmin}} \left\| \underline{\mathbf{P}}_1^{(n)} \circledast \left(\underline{\mathbf{Y}}_1^{(n)} - \mathbf{U}^{(n)} (\mathbf{W}_1^{(n)})^{\top} \right) \right\|_F^2, \tag{6.115}$$

for $n = 1, 2, \ldots, N$.

We know that for given $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{a \times b}$, $\|\mathbf{M} - \mathbf{N}\|_F \geq \operatorname{abs}(\|\mathbf{M}\|_F - \|\mathbf{N}\|_F) \geq \|\mathbf{M}\|_F - \|\mathbf{N}\|_F$. Accordingly, we have

$$\left\| \underline{\mathbf{P}}_1^{(n)} \circledast (\mathbf{U}_1^{(n)} (\mathbf{W}_1^{(n)})^{\top}) \right\|_F \leq 2 \left\| \underline{\mathbf{P}}_1^{(n)} \circledast \underline{\mathbf{Y}}_1^{(n)} \right\|_F < +\infty, \tag{6.116}$$

It is therefore that

$$\sum_{i=1}^{I_n} \left\| \underline{\mathbf{P}}_{1,i}^{(n)} \mathbf{W}_1^{(n)} (\mathbf{u}_i^{(n)})^{\top} \right\|_2^2 < +\infty, \tag{6.117}$$

where $\underline{\mathbf{P}}_{1,i}^{(n)} = \operatorname{diag}\left(\underline{\mathbf{P}}_1^{(n)}(i,:) \right)$ and $\mathbf{u}_i^{(n)}$ is the i-th row of $\mathbf{U}_1^{(n)}$. Since $\{\mathbf{U}_0^{(n)}\}_{n=1}^{N}$ are initialized by full rank and bounded matrices and $\mathbf{u}_1$ is bounded, $\mathbf{W}_1^{(n)}$ is a full column rank matrix. Under the Assumption (A-3), the null space of $\underline{\mathbf{P}}_{1,i}^{(n)} \mathbf{W}_1^{(n)}$ admits only $\mathbf{0}$ as an element. As a result, $\left\| \mathbf{u}_i^{(n)} \right\|_2 < +\infty, i = 1, 2, \ldots, I_n$ and hence $\mathbf{U}_1^{(n)}$ is bounded.

The induction step. Assume that $\{\boldsymbol{\mathcal{U}}_i\}_{i=1}^{k}$ generated by ACP are bounded at time $t = k > 1$, we will prove that at $t = k + 1$, $\boldsymbol{\mathcal{U}}_{k+1}$ is also bounded.

The recursive rule for updating $\mathbf{U}_{k+1}^{(n)}$ is given by

$$\mathbf{U}_{k+1}^{(n)} = \mathbf{U}_k^{(n)} + \Delta\underline{\mathbf{Y}}_{k+1}^{(n)}\big(\mathbf{V}_{k+1}^{(n)}\big)^{\top}, \tag{6.118}$$

where

$$\Delta\underline{\mathbf{Y}}_{k+1}^{(n)} = \underline{\mathbf{P}}_{k+1}^{(n)} \circledast \Big(\underline{\mathbf{Y}}_{k+1}^{(n)} - \mathbf{U}_{\tau}^{(n)}\big(\mathbf{W}_{k+1}^{(n)}\big)^{\top}\Big), \tag{6.119a}$$

$$\mathbf{V}_{k+1}^{(n)} = \big(\mathbf{S}_{k+1}^{(n)}\big)^{-1}, \tag{6.119b}$$

$$\mathbf{S}_{k+1}^{(n)} = \beta\mathbf{S}_k^{(n)} + \big(\mathbf{W}_{k+1}^{(n)}\big)^{\top}\mathbf{W}_{k+1}^{(n)}, \tag{6.119c}$$

$$\mathbf{W}_{k+1}^{(n)} = \Big(\bigodot_{i=1,i\neq n}^{N} \mathbf{U}_k^{(i)}\Big) \odot \mathbf{u}_{k+1}^{\top}, \tag{6.119d}$$

Since $\{\mathbf{U}_k^{(n)}\}_{n=1}^{N}$ are assumed to be bounded and $\mathbf{u}_{k+1}$ is bounded, we obtain that $\mathbf{W}_{k+1}^{(n)}$ and $\Delta\underline{\mathbf{Y}}_{k+1}^{(n)}$ are bounded. Moreover, $\mathbf{S}_{k+1}^{(n)}$ can be recursively expressed by

$$\mathbf{S}_{k+1}^{(n)} = \lambda\mathbf{S}_k^{(n)} + \sum_i \mathbf{w}_i\mathbf{w}_i^{\top}, \tag{6.120}$$

where $\mathbf{w}_i$ is the i-th row of $\mathbf{W}_{k+1}^{(n)}$. Thanks to Sherman-Morrison formula and the initial case $\mathbf{S}_0^{(n)} = \delta_n\mathbf{I}$, $\mathbf{S}_{k+1}^{(n)}$ is a positive definite and invertible matrix and $\mathbf{V}_{k+1}^{(n)}$ is always existent (i.e. inverse of the rank 1 update $\mathbf{S}_{k+1}^{(n)}$). In addition, for any positive definite and invertible matrix $\mathbf{M} \in \mathbb{R}^{r \times r}$, we have

$$\|\mathbf{M}\|_F \leq \sqrt{r}\|\mathbf{M}\|_2 = \sqrt{r}\sigma_{\max}(\mathbf{M}), \text{ and } \big\|\mathbf{M}^{-1}\big\|_2 = \frac{1}{\sigma_{\min}(\mathbf{M})} < +\infty, \tag{6.121}$$

where $\sigma_{\max}(\mathbf{M})$ and $\sigma_{\min}(\mathbf{M})$ are the largest and smallest singular value of $\mathbf{M}$, respectively. Accordingly, we obtain

$$\big\|\mathbf{V}_{k+1}^{(n)}\big\|_F \leq \sqrt{r_{\text{CP}}}\big\|\mathbf{V}_{k+1}^{(n)}\big\|_2 = \frac{\sqrt{r_{\text{CP}}}}{\sigma_{\min}\big(\mathbf{S}_{k+1}^{(n)}\big)}. \tag{6.122}$$

The lower bound on the minimum singular value of $\mathbf{S}_{k+1}^{(n)}$ is specified by the following proposition.

Proposition 13 (Theorem 1 [348] and Theorem 2.1 [349]) *Let* $\mathbf{A}$ *be an $r \times r$ symmetric matrix with positive eigenvalues $\sigma_1(\mathbf{A}) \geq \sigma_2(\mathbf{A}) \geq \cdots \geq \sigma_r(\mathbf{A}) > 0$. If $\mathbf{w}$ is an r-dimensional column vector and $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{w}\mathbf{w}^\top$, we always have*

$$\sigma_r(\mathbf{A}) \leq \sigma_r(\hat{\mathbf{A}}) \leq \sigma_{r-1}(\mathbf{A}) \leq \sigma_{r-1}(\hat{\mathbf{A}}) \leq \dots$$
$$\leq \sigma_1(\mathbf{A}) \leq \sigma_1(\hat{\mathbf{A}}) \leq \sigma_1(\mathbf{A}) + \|\mathbf{w}\|_2^2. \tag{6.123}$$

Accordingly, we have

$$\sigma_{\min}\big(\mathbf{S}_{k+1}^{(n)}\big) \geq \lambda \sigma_{\min}\big(\mathbf{S}_k^{(n)}\big) \geq \lambda^2 \sigma_{\min}\big(\mathbf{S}_{k-1}^{(n)}\big) \geq \dots$$
$$\geq \lambda^{k+1} \sigma_{\min}\big(\mathbf{S}_0^{(n)}\big) = \lambda^{k+1}\delta_n \geq \delta_n. \tag{6.124}$$

The last inequality is when the forgetting factor $\lambda = 1$. As a result, we obtain

$$\Big\|\mathbf{V}_{k+1}^{(n)}\Big\|_F \leq \sqrt{r_{\mathrm{CP}}}\,\delta_n^{-1} < +\infty. \tag{6.125}$$

It implies that $\mathbf{V}_{k+1}^{(n)}$ is bounded. Therefore, $\mathbf{U}_{k+1}$ is bounded, thanks to the rule (6.118).

Proposition 14 *The surrogate $g_t(.)$ is a Lipschitz function.*

Proof. First, we exploit that $g_{t+1}\big(\mathbf{U}_{t+1}^{(n)}\big) \leq g_{t+1}\big(\mathbf{U}_t^{(n)}\big)\forall\, t$ due to $\mathbf{U}_{t+1}^{(n)} = \arg\min g_{t+1}\big(\mathbf{U}^{(n)}\big)$ and hence

$$g_t\big(\mathbf{U}_{t+1}^{(n)}\big) - g_t\big(\mathbf{U}_t^{(n)}\big) = g_t\big(\mathbf{U}_{t+1}^{(n)}\big) - g_{t+1}\big(\mathbf{U}_{t+1}^{(n)}\big) + g_{t+1}\big(\mathbf{U}_{t+1}^{(n)}\big) - g_t\big(\mathbf{U}_t^{(n)}\big)$$
$$\leq \Big(g_t\big(\mathbf{U}_{t+1}^{(n)}\big) - g_{t+1}\big(\mathbf{U}_{t+1}^{(n)}\big)\Big) - \Big(g_t\big(\mathbf{U}_t^{(n)}\big) - g_{t+1}\big(\mathbf{U}_t^{(n)}\big)\Big)$$
$$\overset{\Delta}{=} d_t\big(\mathbf{U}_{t+1}^{(n)}\big) - d_t\big(\mathbf{U}_t^{(n)}\big), \tag{6.126}$$

where $d_t(\mathbf{U}) = g_t(\mathbf{U}) - g_{t+1}(\mathbf{U})$. The derivative of $d_t(\mathbf{U}^{(n)})$ is then given by

$$\frac{\partial d_t\big(\mathbf{U}^{(n)}\big)}{\partial \mathbf{U}^{(n)}} = \mathbf{U}^{(n)}\left(\frac{\mathbf{A}_t}{t} - \frac{\mathbf{A}_{t+1}}{t+1}\right) + \left(\frac{\mathbf{B}_t}{t} - \frac{\mathbf{B}_{t+1}}{t+1}\right), \tag{6.127}$$

where $\mathbf{A}_t = \sum_{\tau=1}^t \beta^{t-\tau}\big(\mathbf{W}_\tau^{(n)}\big)^\top \mathbf{W}_\tau^{(n)}$, $\mathbf{B}_t = \sum_{\tau=1}^t \beta^{t-\tau}\big(\underline{\mathbf{P}}_\tau^{(n)} \circledast \underline{\mathbf{X}}_\tau^{(n)}\big)\mathbf{W}_\tau^{(n)}$. Accordingly, we have

$$\left\|\frac{\partial d_t\big(\mathbf{U}^{(n)}\big)}{\partial \mathbf{U}^{(n)}}\right\|_F \leq \big\|\mathbf{U}^{(n)}\big\|_F \left\|\frac{\mathbf{A}_t}{t} - \frac{\mathbf{A}_{t+1}}{t+1}\right\|_F + \left\|\frac{\mathbf{B}_t}{t} - \frac{\mathbf{B}_{t+1}}{t+1}\right\|_F, \tag{6.128}$$

thanks to the following inequality $\|\mathbf{MN}\|_F \leq \|\mathbf{M}\|_F \|\mathbf{N}\|_F$ for all $\mathbf{M}, \mathbf{N}$. It implies that the function $d_t(\mathbf{U}^{(n)})$ is Lipschitz, i.e.,

$$g_t(\mathbf{U}_{t+1}^{(n)}) - g_t(\mathbf{U}_t^{(n)}) \leq d_t(\mathbf{U}_{t+1}^{(n)}) - d_t(\mathbf{U}_t^{(n)}) \leq c_n \|\mathbf{U}_{t+1}^{(n)} - \mathbf{U}_t^{(n)}\|_F, \qquad (6.129)$$

where the Lipschitz constant $c_n = \mathcal{O}(1/t)$ is given by $c_n = \kappa \|\frac{\mathbf{A}_t}{t} - \frac{\mathbf{A}_{t+1}}{t+1}\|_F + \|\frac{\mathbf{B}_t}{t} - \frac{\mathbf{B}_{t+1}}{t+1}\|_F$, where $\|\mathbf{U}^{(n)}\|_F \leq \kappa$ is the upper bound for $\|\mathbf{U}^{(n)}\|_F$.

In parallel, the surrogate $g_t(\mathcal{U})$ is a multi-convex function because of its quadratic form. It is therefore that

$$g_t(\mathbf{U}_{t+1}^{(n)}) - g_t(\mathbf{U}_t^{(n)}) \geq m_n \|\mathbf{U}_{t+1}^{(n)} - \mathbf{U}_t^{(n)}\|_F^2, \qquad (6.130)$$

where m_n is a positive number. From (6.129) and (6.130), we obtain the following nice corollary:

Corollary 3 *The asymptotic variation of $\mathcal{U}_t$ is given by*

$$\|\mathbf{U}_{t+1}^{(n)} - \mathbf{U}_t^{(n)}\|_F = \mathcal{O}(1/t). \qquad (6.131)$$

6.6.1.2 Step II

We then prove that the nonnegative sequence $\{g_t(\mathcal{U}_t)\}_{t=1}^{\infty}$ converges almost surely where $\{\mathcal{U}_t\}_{t=1}^{\infty}$ is generated by our ACP algorithm.

Convergence of $\{g_t(\mathcal{U}_t)\}_{t=1}^{\infty}$ can be stated in the following proposition:

Proposition 15 *Let $\{\mathcal{U}_t\}_{t=1}^{\infty}$ be a sequence of solutions generated by ACP, the sequence $\{g_t(\mathcal{U}_t)\}_{t=1}^{\infty}$ converges almost surely, i.e.,*

$$\sum_{t=1}^{\infty} \left| \mathbb{E}\left[g_{t+1}(\mathcal{U}_{t+1}) - g_t(\mathcal{U}_t) | \mathcal{F}_t \right] \right| < +\infty \quad a.s., \qquad (6.132)$$

where $\{\mathcal{F}_t\}_{t>0}$ is the filtration of the past estimations at time instant t.

Proof. We begin with the expression

$$g_{t+1}(\mathcal{U}_t) = \frac{1}{t+1} \sum_{\tau=1}^{t+1} \beta^{t+1-k} \tilde{\ell}(\mathcal{U}_t, \mathcal{P}_\tau, \mathcal{Y}_\tau, \mathbf{u}_\tau)$$

$$= \frac{1}{t+1} \left(\tilde{\ell}(\mathcal{U}_t, \mathcal{P}_{t+1}, \mathcal{Y}_{t+1}, \mathbf{u}_{t+1}) + \beta t g_t(\mathcal{U}_t) \right)$$

$$= \frac{\tilde{\ell}(\mathcal{U}_t, \mathcal{P}_{t+1}, \mathcal{Y}_{t+1}, \mathbf{u}_{t+1})}{t+1} + \frac{t(\beta-1)}{t+1} g_t(\mathcal{U}_t) + \frac{t}{t+1} g_t(\mathcal{U}_t). \qquad (6.133)$$

where $\ell(\mathcal{U}, \mathcal{P}, \mathcal{Y}) = \min_{\mathbf{u}} \tilde{\ell}(\mathcal{U}, \mathcal{P}, \mathcal{Y}, \mathbf{u})$. We then have

$$\frac{g_t(\mathcal{U}_t) - f_t(\mathcal{U}_t)}{t+1} = \left(g_t(\mathcal{U}_t) - \frac{t}{t+1} g_t(\mathcal{U}_t) \right) - \frac{f_t(\mathcal{U}_t)}{t+1}$$

$$= g_t(\mathcal{U}_t) - g_{t+1}(\mathcal{U}_{t+1}) + \frac{\tilde{\ell}(\mathcal{U}_t, \mathcal{P}_{t+1}, \mathcal{Y}_{t+1}, \mathbf{u}_{t+1}) - f_t(\mathcal{U}_t)}{t+1}$$

$$+ \underbrace{g_{t+1}(\mathcal{U}_{t+1}) - g_{t+1}(\mathcal{U}_t)}_{\leq 0} + \underbrace{\frac{t(\beta - 1)}{t+1} g_t(\mathcal{U}_t)}_{\leq 0}$$

$$\leq g_t(\mathcal{U}_t) - g_{t+1}(\mathcal{U}_{t+1}) + \frac{\tilde{\ell}(\mathcal{U}_t, \mathcal{P}_{t+1}, \mathcal{Y}_{t+1}, \mathbf{u}_{t+1}) - f_t(\mathcal{U}_t)}{t+1}, \qquad (6.134)$$

because $0 < \beta \leq 1$ and $g_{t+1}(\mathcal{U}_{t+1}) \leq g_{t+1}(\mathcal{U}_t)$ for all t due to $\mathcal{U}_{t+1} = \operatorname{argmin} g_{t+1}(\mathcal{U})$.

Moreover, we know that $\mathbf{u}_{t+1} = \operatorname{argmin}_{\mathbf{u}} \tilde{\ell}(\mathcal{U}_t, \mathcal{P}_{t+1}, \mathcal{Y}_{t+1}, \mathbf{u})$, so

$$\ell(\mathcal{U}_t, \mathcal{P}_{t+1}, \mathcal{Y}_{t+1}) = \tilde{\ell}(\mathcal{U}_t, \mathcal{P}_{t+1}, \mathcal{Y}_{t+1}, \mathbf{u}_{t+1}).$$

Accordingly, we obtain the following inequality

$$g_{t+1}(\mathcal{U}_{t+1}) - g_t(\mathcal{U}_t) \leq \frac{\ell(\mathcal{U}_t, \mathcal{P}_{t+1}, \mathcal{Y}_{t+1}) - f_t(\mathcal{U}_t)}{t+1} - \frac{g_t(\mathcal{U}_t) - f_t(\mathcal{U}_t)}{t+1}. \qquad (6.135)$$

Moreover $f_t(\mathcal{U}_t) \leq g_t(\mathcal{U}_t)$ for all t, we obtain

$$g_{t+1}(\mathcal{U}_{t+1}) - g_t(\mathcal{U}_t) \leq \frac{\ell(\mathcal{U}_t, \mathcal{P}_{t+1}, \mathcal{Y}_{t+1}) - f_t(\mathcal{U}_t)}{t+1}. \qquad (6.136)$$

Taking the expectation of (6.136) conditioned by $\mathcal{F}_t$ results in

$$\mathbb{E}\big[g_{t+1}(\mathcal{U}_{t+1}) - g_t(\mathcal{U}_t)|\mathcal{F}_t\big] \leq \frac{f(\mathcal{U}_t) - f_t(\mathcal{U}_t)}{t+1}, \qquad (6.137)$$

where $f(\mathcal{U})$ be the expected cost function, i.e., $f(\mathcal{U}) = \lim_{t \to \infty} f_t(\mathcal{U})$ and $\mathbb{E}\big[\ell(\mathcal{U}, \mathcal{P}_{t+1}, \mathcal{Y}_{t+1})\big] = f(\mathcal{U})$, for all $\mathcal{U}$. Now, let us define the indicator function δ_t as follows

$$\delta_t \overset{\Delta}{=} \begin{cases} 1 & \text{if } \mathbb{E}\big[g_{t+1}(\mathcal{U}_{t+1}) - g_t(\mathcal{U}_t)|\mathcal{F}_t\big] > 0, \\ 0 & \text{otherwise.} \end{cases} \qquad (6.138)$$

Accordingly, we have

$$\mathbb{E}\Big[\delta_t \mathbb{E}\big[g_{t+1}(\mathcal{U}_{t+1}) - g_t(\mathcal{U}_t)|\mathcal{F}_t\big]\Big] \leq \mathbb{E}\Big[\sqrt{t}\big(f(\mathcal{U}_t) - f_t(\mathcal{U}_t)\big)\Big] \frac{1}{\sqrt{t}(t+1)}. \qquad (6.139)$$

Under the given assumptions that variables are bounded, we exploit that the set of loss functions $\{\ell(\boldsymbol{U}_t, \boldsymbol{P}_t, \boldsymbol{X}_t)\}_{t \geq 1}$ is $\mathbb{P}$-Donsker [126]. As a result, the centered and scaled version of $f_t(\boldsymbol{U}_t)$ satisfies the following proposition: $\mathbb{E}\left[\sqrt{t}\big(f(\boldsymbol{U}_t) - f_t(\boldsymbol{U}_t)\big)\right] = \mathcal{O}(1)$, thanks to the Donsker theorem [126, Section 19.2].

We then consider the convergence of the sum $\sum_{t=1}^{+\infty} \frac{1}{\sqrt{t}(t+1)}$. In particular, the Cauchy-MacLaurin integral test [133] is applied for examining the convergence, that is, $\int_{t=1}^{+\infty} \frac{1}{\sqrt{t}(t+1)} dt = \frac{\pi}{4} < \infty$. Accordingly, $\left\{\frac{1}{\sqrt{t}(t+1)}\right\}_{t>0}$ converges. Therefore, we obtain

$$\sum_{t=1}^{\infty} \mathbb{E}\left[\delta\mathbb{E}\big[g_{t+1}(\boldsymbol{U}_{t+1}) - g_t(\boldsymbol{U}_t)|\mathcal{F}_t\big]\right] < \infty. \tag{6.140}$$

According to quasi-martingale theorem [343, Theorem 9.4 & Proposition 9.5], we can conclude that $\{g_t(\boldsymbol{U}_t)\}_{t=1}^{\infty}$ converges almost surely, i.e.,

$$\sum_{t=1}^{\infty} \mathbb{E}\big[g_{t+1}(\boldsymbol{U}_{t+1}) - g_t(\boldsymbol{U}_t)|\mathcal{F}_t\big] < \infty. \tag{6.141}$$

We complete the proof.

Stage III

The last stage contains two main steps: (i) we first indicate that the empirical cost function $f_t(\boldsymbol{U})$ is not only continuously differentiable, but also Lipschitz; (ii) we then prove $\{f_t(\boldsymbol{U}_t)\}_{t=1}^{\infty}$ and $\{g_t(\boldsymbol{U}_t)\}_{t=1}^{\infty}$ converge to the same limit. As a result, the derivative of $f_t(\boldsymbol{U})$ equals to that of $g_t(\boldsymbol{U})$ when $t \to \infty$, thanks to the first-order Taylor approximation. Since $\boldsymbol{U}_t$ is the minimizer of $g_t(\boldsymbol{U})$, the derivative $\nabla f_t(\boldsymbol{U}) \to 0$ *a.s.*

To begin with, we provide the following proposition which is a corollary of Theorem 4.1 in [350]:

Proposition 16 *Consider a continuous function $f : \mathcal{V} \times \mathcal{U} \to \mathbb{R}$. Suppose that $\forall \mathbf{u} \in \mathcal{U}$, the function $f(., \mathbf{u})$ is differentiable and $\nabla_{\mathbf{v}} f(\mathbf{v}, \mathbf{u})$ is continuous on $\mathcal{V} \times \mathcal{U}$. If $g(\mathbf{v})$ be the function derived from $g(\mathbf{v}) = \min_{\mathbf{u} \in \mathcal{U}} f(\mathbf{v}, \mathbf{u})$, then $g(\mathbf{v})$ is also differentiable. In addition, if $\mathbf{u}^* = \operatorname{argmin}_{\mathbf{u} \in \mathcal{U}} f(\mathbf{v}, \mathbf{u})$ be unique, $\nabla g(\mathbf{v}) = \nabla_{\mathbf{v}} f(\mathbf{v}, \mathbf{u}^*)$, $\forall \mathbf{v} \in \mathcal{V}$.*

Proof. Its proof is already provided in [350, Theorem 4.1].

Accordingly, we derive the following proposition to verify the differentiable property of $\ell(\boldsymbol{U}, \boldsymbol{P}_t, \boldsymbol{X}_t)$ at time t.

Corollary 4 *Given an incomplete observation $\mathcal{P}_t \circledast \mathcal{X}_t$ and the past estimation $\mathcal{U}$, let $\mathbf{u}_t^*$ be the minimizer of the summand $\tilde{\ell}(\mathcal{U}, \mathcal{P}_t, \mathcal{X}_t, \mathbf{u})$*

$$\mathbf{u}_t^* = \operatorname*{argmin}_{\mathbf{u}_t \in \mathbb{R}^r} \left\| \mathcal{P}_t \circledast \left(\mathcal{Y}_t - [\![\{\mathbf{U}^{(n)}\}_{n=1}^{N-1}, \mathbf{u}_t^{(N)} [\![\right) \right\|_F^2. \tag{6.142}$$

We obtain that $\ell(\mathcal{U}, \mathcal{P}_t, \mathcal{X}_t) = \min_{\mathbf{u}_t} \tilde{\ell}(\mathcal{U}, \mathcal{P}_t, \mathcal{X}_t, \mathbf{u}_t)$ is a continuously differentiable function and its partial derivative w.r.t. $\mathbf{U}^{(n)}$ is given by

$$\frac{\partial \ell(\mathcal{U}, \mathcal{P}_t, \mathcal{X}_t)}{\partial \mathbf{U}^{(n)}} = 2\underline{\mathbf{P}}_t^{(n)} \circledast \left(\underline{\mathbf{Y}}_t^{(n)} - \mathbf{U}^{(n)} \left(\mathbf{W}_t^*\right)^\top \right) \mathbf{W}_t^*, \tag{6.143}$$

where $\mathbf{W}_t^* = \left(\overset{N}{\underset{i=1, i\neq n}{\odot}} \mathbf{U}_{t-1}^{(i)} \right) \odot (\mathbf{u}_t^*)^\top. \tag{6.144}$

As a result, the empirical cost function $f_t(\mathcal{U}) = \frac{1}{t}\beta^{t-\tau} \sum_{\tau=1}^t \ell(\mathcal{U}, \mathcal{P}_\tau, \mathcal{Y}_\tau)$ is continuously differentiable. Applying the same augments in Proposition 14, we also have

$$\left\| \frac{\partial \bar{f}_t}{\partial \mathbf{U}^{(n)}} \right\|_F \leq \left\| \mathbf{U}^{(n)} \right\|_F \left\| \frac{\bar{\mathbf{A}}_t^{(n)}}{t} - \frac{\bar{\mathbf{A}}_{t+1}^{(n)}}{t+1} \right\|_F + \left\| \frac{\bar{\mathbf{B}}_t^{(n)}}{t} - \frac{\bar{\mathbf{B}}_{t+1}^{(n)}}{t+1} \right\|_F, \tag{6.145}$$

where $\bar{f}_t(\mathbf{U}^{(n)}) = f_t(\mathbf{U}^{(n)}) - f_{t+1}(\mathbf{U}^{(n)})$, $\bar{\mathbf{A}}_t^{(n)} = \sum_{\tau=1}^t \beta^{t-\tau} (\mathbf{W}_\tau^*)^\top \mathbf{W}_\tau^*$, and $\bar{\mathbf{B}}_t^{(n)} = \sum_{\tau=1}^t \beta^{t-\tau} (\underline{\mathbf{P}}_{\underline{\tau}}^{(n)} \circledast \underline{\mathbf{Y}}_{\underline{\tau}}^{(n)}) \mathbf{W}_\tau^*$. All terms in the right side are bounded, the partial derivative $\bar{f}_t(\mathcal{U})$ w.r.t. $\mathbf{U}^{(n)}$ is bounded and hence

$$f_t(\mathbf{U}_{t+1}^{(n)}) - f_t(\mathbf{U}_t^{(n)}) \leq d_n \left\| \mathbf{U}_{t+1}^{(n)} - \mathbf{U}_t^{(n)} \right\|_F, \tag{6.146}$$

where d_n is the deterministic positive number. It implies that $f_t(.)$ is Lipschitz continuous.

Now, we indicate that the nonnegative sequence $\left\{ (g_t(\mathcal{U}_t) - f_t(\mathcal{U}_t)) \frac{1}{t+1} \right\}$ converges almost surely. We prove that the empirical cost function $\{f_t(\mathcal{U}_t)\}_{t=1}^\infty$ and its surrogate $\{g_t(\mathcal{U}_t)\}_{t=1}^\infty$ converge to the same limit by showing

$$\sum_{t=1}^\infty g_t(\mathcal{U}_t) - f_t(\mathcal{U}_t) < +\infty. \tag{6.147}$$

According to (6.174), we recall the following inequality

$$\frac{g_t(\mathcal{U}_t) - f_t(\mathcal{U}_t)}{t+1} \leq g_t(\mathcal{U}_t) - g_{t+1}(\mathcal{U}_{t+1}) + \frac{\ell(\mathcal{U}_t, \mathcal{P}_{t+1}, \mathcal{Y}_{t+1}) - f_t(\mathcal{U}_t)}{t+1}. \tag{6.148}$$

To examine the convergence of the right side of (6.148), we exploit the following facts: (i) The convergence of $\mathbb{E}\big[g_t(\mathcal{U}_t) - g_{t+1}(\mathcal{U}_{t+1})|\mathcal{F}_t\big]$ is already provided in Proposition 2, and (ii) The second term also converges, thanks to the convergence of $\mathbb{E}[f(\mathbf{U}_t) - f_t(\mathbf{U}_t)]/(t+1)$ and $\mathbb{E}[\ell(\mathcal{U}_t, \mathcal{P}, \mathcal{X})] = f(\mathcal{U}_t)$ for all t.

Accordingly, we have that $\big\{\big(g_t(\mathcal{U}_t) - f_t(\mathcal{U}_t)\big)\frac{1}{t+1}\big\}$ converges

$$\sum_{t=0}^{\infty} \Big(g_t(\mathcal{U}_t) - f_t(\mathcal{U}_t)\Big)\frac{1}{t+1} < \infty. \tag{6.149}$$

Since both $g_t(\mathcal{U})$ and $f_t(\mathcal{U})$ are Lipschitz continuous, there always exist a constant $L > 0$ such that

$$\big|(g_{t+1}(\mathcal{U}_{t+1}) - f_{t+1}(\mathcal{U}_{t+1})) - (g_t(\mathcal{U}_t) - f_t(\mathcal{U}_t))\big| \le L\big\|\mathcal{U}_{t+1} - \mathcal{U}_t\big\|_F. \tag{6.150}$$

In addition, the real sequence $\big\{\frac{1}{t+1}\big\}_{t \ge 0}$ diverges, i.e., $\sum_{t=0}^{\infty} \frac{1}{t+1} = +\infty$. It implies that $\sum_{t=0}^{\infty} g_t(\mathcal{U}_t) - f_t(\mathcal{U}_t) < \infty$, thanks to [351, Lemma A.5]. It results in $g_t(\mathcal{U}_t) \overset{a.s}{\to} f_t(\mathcal{U}_t), t \to \infty$.

In parallel, $g_t(\mathcal{U})$ is the surrogate function of $f_t(\mathcal{U})$, we always have

$$g_t(\mathcal{U} + a_\tau \mathcal{V}) \ge f_t(\mathcal{U} + a_\tau \mathcal{V}), \tag{6.151}$$

for all $\mathcal{V}$ and the nonnegative sequence $\{a_\tau\}$. For short, let us denote $g_t\big(\mathbf{U}^{(n)}\big) \overset{\Delta}{=} g_t(\mathcal{U})$ and $f_t\big(\mathbf{U}^{(n)}\big) \overset{\Delta}{=} f_t(\mathcal{U})$ when the remaining loading factors are fixed. With respect to $\mathbf{U}^{(n)}$, the inequality (6.151) becomes

$$g_t\big(\mathbf{U}^{(n)} + a_\tau \mathbf{V}^{(n)}\big) \ge f_t\big(\mathbf{U}^{(n)} + a_\tau \mathbf{V}^{(n)}\big). \tag{6.152}$$

Thanks to Taylor's theorem, taking the linear approximation of (6.180) yields

$$g_t\big(\mathbf{U}_t^{(n)}\big) + \operatorname{tr}\big[a_\tau\big(\mathbf{V}^{(n)}\big)^\top \nabla g_t\big(\mathbf{U}_t^{(n)}\big)\big] + o\big(a_\tau \mathbf{V}^{(n)}\big)$$
$$\ge f_t\big(\mathbf{U}_t^{(n)}\big) + \operatorname{tr}\big[a_\tau\big(\mathbf{V}^{(n)}\big)^\top \nabla f_t\big(\mathbf{U}_t^{(n)}\big)\big] + o\big(a_\tau \mathbf{V}^{(n)}\big). \tag{6.153}$$

When $t \to \infty$, we have $g_t\big(\mathbf{U}_t^{(n)}\big) = f_t\big(\mathbf{U}_t^{(n)}\big)$ as proved in Lemma 1 and hence

$$\operatorname{tr}\big[a_\tau\big(\mathbf{V}^{(n)}\big)^\top \nabla g_t\big(\mathbf{U}_t^{(n)}\big)\big] \ge \operatorname{tr}\big[a_\tau\big(\mathbf{V}^{(n)}\big)^\top \nabla f_t\big(\mathbf{U}_t^{(n)}\big)\big]. \tag{6.154}$$

Since the above inequality must hold for all $\mathbf{V}^{(n)}$ and $\{a_\tau\}$, we obtain

$$\nabla g_t\big(\mathbf{U}_t^{(n)}\big) - \nabla f_t\big(\mathbf{U}_t^{(n)}\big) \to 0, \quad \text{when } t \to \infty. \tag{6.155}$$

Because $\mathcal{U}_t$ is the minimizer of $g_t(\mathcal{U})$, we derive $\nabla f_t(\mathcal{U}_t) \to 0$ *a.s*. It ends the proof.

6.6.2 Appendix B: Proof of Lemma 11

Boundedness: $\{\mathbf{D}_t, \mathcal{O}_t, \mathbf{u}_t\}_{t=1}^{\infty}$ are uniformly bounded.

At each time $t > 0$, the outlier $\mathcal{O}_t$ and the coefficient vector $\mathbf{u}_t$ are derived from the minimization (7) in the main manuscript. Accordingly, we always have

$$\tilde{\ell}\big(\mathbf{D}_{t-1}, \mathcal{P}_t, \mathcal{Y}_t, \mathcal{O}_t, \mathbf{u}_t\big) \leq \tilde{\ell}\big(\mathbf{D}_{t-1}, \mathcal{P}_t, \mathcal{Y}_t, \mathbf{0}, \mathbf{0}\big). \tag{6.156}$$

It is therefore that

$$\|\mathcal{O}_t\|_1 + \frac{\rho}{2}\big\|\mathcal{P}_t \circledast (\mathcal{Y}_t - \mathcal{O}_t - \mathcal{H}_{t-1} \times_N \mathbf{u}_t)\big\|_F^2 \leq \frac{\rho}{2}\|\mathcal{P}_t \circledast \mathcal{Y}_t\|_F^2. \tag{6.157}$$

Due to the two facts that $\|\mathbf{M}\|_F + \|\mathbf{N}\|_F \geq \|\mathbf{M} - \mathbf{N}\|_F \geq \|\mathbf{M}\|_F - \|\mathbf{N}\|_F$, and $\|\mathbf{M}\|_F \leq \|\mathbf{M}\|_1$ [9], we then obtain

$$\|\mathcal{O}_t\|_F \leq \|\mathcal{O}_t\|_1 \leq \frac{\rho}{2}\|\mathcal{P}_t \circledast \mathcal{Y}_t\|_F^2 \leq \frac{\rho}{2}M_x^2 < \infty, \tag{6.158}$$

$$\big\|\mathbf{P}_t\mathbf{H}_{t-1}\mathbf{u}_t\big\|_2 \leq 2\|\mathcal{P}_t \circledast \mathcal{Y}_t\|_F + \|\mathcal{P}_t \circledast \mathcal{O}_t\|_F < \infty, \tag{6.159}$$

where M_x is the upper bound of $\|\mathcal{Y}_t\|_F$ (see Assumption A1). Thanks to (6.158), $\mathcal{O}_t$ is uniformly bound.

We indicate the bound of the solution $\mathbf{u}_t$ and $\mathbf{D}_t = \big[\mathbf{U}_t^{(1)}, \ldots, \mathbf{U}_t^{(N)}\big]$ by using the mathematical induction.

We first recall that the proposed RACP algorithm begins with N full-rank matrices $\{\mathbf{U}_0^{(n)}\}_{n=1}^N$ and a set of matrices $\mathbf{S}_{0,m}^{(n)} = \delta_n \mathbf{I}$, $m = 1, 2, \ldots, I_n$.

The base case: At $t = 1$, the matrix $\mathbf{H}_0 = \bigodot_{n=1}^N \mathbf{U}_0^{(n)}$ is then full rank, i.e., the null space of $\mathbf{H}_0$ admits only $\mathbf{0}$ as a vector. Accordingly, $\mathbf{u}_1$ is bounded, thanks to (6.159).

To indicate the bound of $\mathbf{U}_1^{(n)}$ for $n = 1, 2, \ldots, N$, we show that each row $\mathbf{u}_{1,m}^{(n)}$ of $\mathbf{U}_1^{(n)}$ is bounded. We first obtain the following inequality

$$\big\|\mathbf{u}_{1,m}^{(n)}\big\|_2 \leq \big\|\mathbf{u}_{0,m}^{(n)}\big\|_2 + \big\|\underline{\mathbf{P}}_{1,m}^{(n)}\big((\underline{\mathbf{x}}_{1,m}^{(n)})^\top - \mathbf{W}_1^{(n)}(\mathbf{u}_{0,m}^{(n)})^\top\big)\big\|_2 \big\|\mathbf{V}_{1,m}^{(n)}\big\|_2. \tag{6.160}$$

In fact, three matrices $\mathbf{W}_{1,m}^{(n)}$, $\mathbf{S}_{1,m}^{(n)}$ and $\mathbf{V}_{1,m}^{(n)}$ for updating $\mathbf{u}_{1,m}^{(n)}$ are bounded due to the bound of $\{\mathbf{U}_0^{(n)}\}_{n=1}^N$. Accordingly, the right hand side of (6.160) is finite, thus $\mathbf{u}_{1,m}^{(n)}$ is bounded for all m. It implies that $\mathbf{U}_1^{(n)}$ is bounded.

The induction step: We assume that $\{\mathbf{U}_i^{(n)}\}_{i=1}^k$ generated by RACP are bounded at time $t = k > 1$, we will prove that at $t = k + 1$, $\mathbf{U}_{k+1}^{(n)}$ is also bounded.

Since $\{\mathbf{U}_k^{(n)}\}_{n=1}^N$ are assumed to be bounded, $\mathbf{u}_{k+1}$ and $\mathbf{W}_{k+1,m}^{(n)}$ are then bounded. In parallel, we exploit that $\mathbf{S}_{k+1,m}^{(n)}$ can be expressed by

$$\mathbf{S}_{k+1,m}^{(n)} = \lambda \mathbf{S}_{\tau,m}^{(n)} + \sum_i p_{-k+1,m}^{(n)}(i) \mathbf{w}_i^\top \mathbf{w}_i, \qquad (6.161)$$

where $\mathbf{w}_i$ is the i-th row of $\mathbf{W}_{k+1,m}^{(n)}$. Thanks to Woodbury matrix identity [352] and $\mathbf{S}_{0,m}^{(n)} = \delta \mathbf{I}$ with $\delta > 0$, we obtain $\mathbf{S}_{k+1,m}^{(n)} > 0$, i.e., $\mathbf{S}_{k+1,m}^{(n)}$ is nonsingular with the smallest eigenvalue $\sigma_{\min}(\mathbf{S}_{k+1,m}^{(n)}) \geq \delta > 0$. Thus $\mathbf{V}_{k+1,m}^{(n)}$ is always existent. For given $\mathbf{M} > 0$, we always have $\|\mathbf{M}\|_F \leq \sqrt{r}\|\mathbf{M}\|_2 = \sqrt{r}\sigma_{\max}(\mathbf{M})$, and $\|\mathbf{M}^{-1}\|_2 = \sigma_{\min}^{-1}(\mathbf{M})$ where $\sigma_{\max}(\mathbf{M})$ and $\sigma_{\min}(\mathbf{M})$ are the largest and smallest eigenvalue of $\mathbf{M}$ [9]. Accordingly, we derive $\|\mathbf{V}_{k+1,m}^{(n)}\|_F \leq \sqrt{r}/\delta < \infty$, i.e., $\mathbf{V}_{k+1,m}^{(n)}$ is bounded. As a result, $\mathbf{u}_{k+1,m}^{(n)}$ is bounded for all $m = 1, 2, \ldots, I_n$. Thanks to the mathematical induction, we can conclude that the solution $\mathbf{U}_t^{(n)}$ generated by RACP is bounded for $t \geq 1$.

Forward Monotonicity: $\tilde{f}_t(\mathbf{D}_{t-1}) \geq \tilde{f}_t(\mathbf{D}_t)$.

We have

$$
\begin{aligned}
&\tilde{f}_t(\mathbf{D}_{t-1}) - \tilde{f}_t(\mathbf{D}_t) \\
&= \begin{cases}
\sum_{n=1}^N \tilde{f}_t\big(\mathbf{U}_{t-1}^{(1)}, \ldots, \mathbf{U}_{t-1}^{(n-1)}, \mathbf{U}_{t-1}^{(n)}, \ldots, \mathbf{U}_{t-1}^{(N)}\big) & \text{[Jacobi]} \\
\quad -\tilde{f}_t\big(\mathbf{U}_{t-1}^{(1)}, \ldots, \mathbf{U}_{t-1}^{(n-1)}, \mathbf{U}_t^{(i)}, \ldots, \mathbf{U}_{t-1}^{(N)}\big) & \\
\sum_{n=1}^N \tilde{f}_t\big(\mathbf{U}_t^{(1)}, \ldots, \mathbf{U}_t^{(n-1)}, \mathbf{U}_{t-1}^{(n)}, \ldots, \mathbf{U}_{t-1}^{(N)}\big) & \text{[Gauss-Seidel]} \\
\quad -\tilde{f}_t\big(\mathbf{U}_t^{(1)}, \ldots, \mathbf{U}_t^{(n-1)}, \mathbf{U}_t^{(n)}, \ldots, \mathbf{U}_{t-1}^{(N)}\big) &
\end{cases}
\qquad (6.162)
\end{aligned}
$$

Recall that $\mathbf{U}_t^{(n)}$ is the minimizer of $\tilde{f}_t\big(\mathbf{U}_{t-1}^{(1)}, \ldots, \mathbf{U}_{t-1}^{(n-1)}, \mathbf{U}, \mathbf{U}_{t-1}^{(n+1)}, \ldots, \mathbf{U}_{t-1}^{(N)}\big)$ if using Jacobi scheme or $\tilde{f}_t\big(\mathbf{U}_t^{(1)}, \ldots, \mathbf{U}_t^{(n-1)}, \mathbf{U}, \mathbf{U}_{t-1}^{(n+1)}, \ldots, \mathbf{U}_{t-1}^{(N)}\big)$ if using Gauss-Seidel scheme. Therefore, we always have

$$\tilde{f}_t\big(\ldots, \mathbf{U}_{t-1}^{(n-1)}, \mathbf{U}_{t-1}^{(n)}, \ldots, \mathbf{U}_{t-1}^{(N)}\big) \geq \tilde{f}_t\big(\ldots, \mathbf{U}_{t-1}^{(n-1)}, \mathbf{U}_t^{(i)}, \ldots, \mathbf{U}_{t-1}^{(N)}\big) \quad \text{[Jacobi]}$$
$$\tilde{f}_t\big(\ldots, \mathbf{U}_{t-1}^{(n-1)}, \mathbf{U}_{t-1}^{(n)}, \ldots, \mathbf{U}_{t-1}^{(N)}\big) \geq \tilde{f}_t\big(\ldots, \mathbf{U}_{t-1}^{(n-1)}, \mathbf{U}_t^{(i)}, \ldots, \mathbf{U}_{t-1}^{(N)}\big) \quad \text{[Gauss-Seidel]}$$

As a result, $\tilde{f}_t(\mathbf{D}_{t-1}) \geq \tilde{f}_t(\mathbf{D}_t)$.

Backward Monotonicity: $\tilde{f}_t(\mathbf{D}_t) \leq \tilde{f}_t(\mathbf{D}_{t+1})$.

Applying the similar argument above, we also obtain $\tilde{f}_t(\mathbf{D}_t) \leq \tilde{f}_t(\mathbf{D}_{t+1})$.

Stability of Estimates: $\|\mathbf{D}_t - \mathbf{D}_{t-1}\|_F = \mathcal{O}(1/t)$.

We first prove that the surrogate $\tilde{f}_t(.)$ w.r.t. each factor is Lipschitz continuous. Since $\mathbf{U}_t^{(n)} = \arg\min \tilde{f}_t(\mathbf{U}^{(n)},.)$, we have $\tilde{f}_t(\mathbf{U}_t^{(n)},.) \le \tilde{f}_t(\mathbf{U}_{t-1}^{(n)},.)\forall t$ and hence

$$\tilde{f}_{t-1}(\mathbf{U}_t^{(n)},.) - \tilde{f}_{t-1}(\mathbf{U}_{t-1}^{(n)},.) \le \left\{ \tilde{f}_{t-1}(\mathbf{U}_t^{(n)},.) - \tilde{f}_t(\mathbf{U}_t^{(n)},.) \right\}$$
$$- \left\{ \tilde{f}_{t-1}(\mathbf{U}_{t-1}^{(n)},.) - \tilde{f}_t(\mathbf{U}_{t-1}^{(n)},.) \right\}. \qquad (6.163)$$

Lets denote the error function $d_t(\mathbf{U}^{(n)},.) = \tilde{f}_{t-1}(\mathbf{U}^{(n)},.) - \tilde{f}_t(\mathbf{U}^{(n)},.)$. We have

$$\nabla d_t(\mathbf{U}^{(n)},.) = \mathbf{U}^{(n)}\left(\frac{\mathbf{A}_{t-1}}{t-1} - \frac{\mathbf{A}_t}{t} \right) + \left(\frac{\mathbf{B}_{t-1}}{t-1} - \frac{\mathbf{B}_t}{t} \right), \qquad (6.164)$$

where $\mathbf{A}_t = \sum_{\tau=1}^{t} \beta^{t-\tau}(\mathbf{W}_\tau^{(n)})^\top \mathbf{W}_\tau^{(n)}, \mathbf{B}_t = \sum_{\tau=1}^{t} \beta^{t-\tau}(\underline{\mathbf{P}}_\tau^{(n)} \circledast (\underline{\mathbf{Y}}_\tau^{(n)} - \underline{\mathbf{O}}_\tau^{(n)}))\mathbf{W}_\tau^{(n)}$. Thanks to the two facts that $\|\mathbf{MN}\|_F \le \|\mathbf{M}\|_F \|\mathbf{N}\|_F$ and $\|\mathbf{M} + \mathbf{N}\|_F \le \|\mathbf{M}\|_F + \|\mathbf{N}\|_F$ [9], we obtain

$$\left\| \nabla d_t(\mathbf{U}^{(n)},.) \right\|_F \le \kappa_U \left\| \frac{\mathbf{A}_{t-1}}{t-1} - \frac{\mathbf{A}_t}{t} \right\|_F + \left\| \frac{\mathbf{B}_{t-1}}{t-1} - \frac{\mathbf{B}_t}{t} \right\|_F = c_n, \qquad (6.165)$$

where κ_U is the upper bound for $\|\mathbf{U}^{(n)}\|_F$. As a result, the error function $d_t(\mathbf{U}^{(n)})$ is Lipschitz with parameter $c_n = \mathcal{O}(1/t)$, i.e.,

$$\tilde{f}_{t-1}(\mathbf{U}_t^{(n)},.) - \tilde{f}_{t-1}(\mathbf{U}_{t-1}^{(n)},.) \le d_t(\mathbf{U}_t^{(n)},.) - d_t(\mathbf{U}_{t-1}^{(n)},.) \le c_n \|\mathbf{U}_t^{(n)} - \mathbf{U}_{t-1}^{(n)}\|_F. \qquad (6.166)$$

Moreover, $\tilde{f}_t(\mathbf{U}^{(n)},.)$ is a m-strongly convex function, i.e.,

$$\tilde{f}_{t-1}(\mathbf{U}_t^{(n)},.) - \tilde{f}_{t-1}(\mathbf{U}_{t-1}^{(n)},.) \ge m \|\mathbf{U}_t^{(n)} - \mathbf{U}_{t-1}^{(n)}\|_F^2.$$

From that, we obtain the asymptotic variation of $\mathbf{U}^{(n)}$ as follows $\|\mathbf{U}_t^{(n)} - \mathbf{U}_{t-1}^{(n)}\|_F \le \frac{c_n}{m} = \mathcal{O}(1/t)$, Therefore, we can conclude that $\sum_{n=1}^{N} \|\mathbf{U}_t^{(n)} - \mathbf{U}_{t-1}^{(n)}\|_F^2 = \|\mathbf{D}_t - \mathbf{D}_{t-1}\|_F^2 = \mathcal{O}(1/t^2)$ or $\|\mathbf{D}_t - \mathbf{D}_{t-1}\|_F = \mathcal{O}(1/t)$.

Stability of Errors: $\left| e_t(\mathbf{D}_t) - e_{t-1}(\mathbf{D}_{t-1}) \right| = \mathcal{O}(1/t)$.

We begin with verifying the differentiable property of the loss function $\ell(\mathbf{D}, \mathcal{P}_t, \mathcal{Y}_t)$ at time t.

Proposition 17 *Given an incomplete observation $\mathcal{P}_t \circledast \mathcal{Y}_t$ and the past estimation of $\mathbf{D}$, let $O_t, \mathbf{u}_t^*$ be the minimizer of $\tilde{\ell}(\mathbf{D}, \mathcal{P}_t, \mathcal{Y}_t, O, \mathbf{u})$, i.e.,*

$$\{\mathbf{u}_t^*, O_t^*\} = \underset{\mathbf{u}, O}{\arg\min} \|O\|_1 + \frac{\rho}{2}\big\|\mathcal{P}_t \circledast (\mathcal{Y}_t - O - \mathcal{H} \times_N \mathbf{u})\big\|_F^2. \quad (6.167)$$

where $\mathcal{H} = \mathcal{I} \prod_{n=1}^{N-1} \times_n \mathbf{U}^{(n)}$. We obtain that $\ell(\mathbf{D}, \mathcal{P}_t, \mathcal{Y}_t) = \min_{\mathbf{u}, O} \tilde{\ell}(\mathbf{D}, \mathcal{P}_t, \mathcal{Y}_t, O, \mathbf{u})$ is a continuously differentiable function and its partial derivative w.r.t. $\mathbf{U}^{(n)}$ is given by

$$\frac{\partial \ell(\mathbf{D}, \mathcal{P}_t, \mathcal{Y}_t)}{\partial \mathbf{U}^{(n)}} = 2\underline{\mathbf{P}}_t^{(n)} \circledast \Big(\underline{\mathbf{Y}}_t^{(n)} - O_t^{(n)} - \mathbf{U}^{(n)}(\bar{\mathbf{W}}_t^{(n)})^\top\Big)\bar{\mathbf{W}}_t^{(n)},$$

$$(6.168)$$

where $\displaystyle \bar{\mathbf{W}}_t^{(n)} = \left(\overset{N-1}{\underset{i=1, i \neq n}{\bigodot}} \mathbf{U}_{t-1}^{(i)}\right) \odot (\mathbf{u}_t^*)^\top.$ $\quad (6.169)$

Proof. The result follows intermediately Theorem 4.1 in [350, page 237].

Accordingly, the sum $f_t(\mathbf{D}) = 1/L_t \sum_{\tau=t-L_t+1}^{t} \beta^{t-\tau} \ell(\mathbf{D}, \mathcal{P}_\tau, \mathcal{Y}_\tau)$ is continuously differentiable.

Let us denote $\bar{f}_t(\mathbf{U}^{(n)}, .) = f_{t-1}(\mathbf{U}^{(n)}, .) - f_t(\mathbf{U}^{(n)}, .)$. Applying the same arguments in subsection I.4, we also obtain

$$\big\|\nabla \bar{f}_t(\mathbf{U}^{(n)}, .)\big\|_F \leq \kappa_U \left\|\frac{\bar{\mathbf{A}}_{t-1}^{(n)}}{t-1} - \frac{\bar{\mathbf{A}}_t^{(n)}}{t}\right\|_F + \left\|\frac{\bar{\mathbf{B}}_{t-1}^{(n)}}{t-1} - \frac{\bar{\mathbf{B}}_t^{(n)}}{t}\right\|_F = d_n, \quad (6.170)$$

where $\bar{\mathbf{A}}_t^{(n)} = \sum_{\tau=1}^{t} \beta^{t-\tau}(\bar{\mathbf{W}}_\tau^{(n)})^\top \bar{\mathbf{W}}_\tau^{(n)}$, and $\bar{\mathbf{B}}_t^{(n)} = \sum_{\tau=1}^{t} \beta^{t-\tau}(\underline{\mathbf{P}}_\tau^{(n)} \circledast (\underline{\mathbf{Y}}_\tau^{(n)} - O_\tau^{(n)}))\bar{\mathbf{W}}_\tau^{(n)}$. Accordingly, $\nabla \bar{f}_t(\mathbf{U}^{(n)}, .)$ is bounded and hence

$$f_t(\mathbf{U}_{t-1}^{(n)}, .) - f_t(\mathbf{U}_t^{(n)}, .) \leq d_n \big\|\mathbf{U}_{t-1}^{(n)} - \mathbf{U}_t^{(n)}\big\|_F. \quad (6.171)$$

It implies that $f_t(.)$ is Lipschitz continuous. Since $\tilde{f}_t(\mathbf{D})$ and $f_t(\mathbf{D})$ are both Lipschitz continuous functions, we then have

$$
\begin{aligned}
\big|e_t(\mathbf{D}_t) - e_{t-1}(\mathbf{D}_{t-1})\big| &= \big|(\tilde{f}_t(\mathbf{D}_t) - f_t(\mathbf{D}_t)) - (\tilde{f}_{t-1}(\mathbf{D}_{t-1}) - f_{t-1}(\mathbf{D}_{t-1}))\big| \\
&\leq \big|\tilde{f}_t(\mathbf{D}_t) - \tilde{f}_t(\mathbf{D}_{t-1})\big| + \big|f_t(\mathbf{D}_t) - f_t(\mathbf{D}_{t-1})\big| \\
&\leq \sum_{n=1}^{N}(c_n + d_n)\big\|\mathbf{U}_{t-1}^{(n)} - \mathbf{U}_t^{(n)}\big\|_F = O(1/t). \quad (6.172)
\end{aligned}
$$

It ends the proof.

6.6.3   Appendix D: Proof of Lemma 12

Detailed Proof: We apply the similar arguments of Proposition 7 in our companion work [30] to prove Lemma 12.

Almost sure convergence of $\{\tilde{f}_t(\mathbf{D}_t)\}_{t=1}^{\infty}$

Main approach: We prove the convergence of the sequence $\tilde{f}_t(\mathbf{D}_t)$ by showing that the stochastic positive process $u_t := \tilde{f}_t(\mathbf{D}_t)$ is a quasi-martingale Fisk. In particular, if the sum of the positive difference of u_t is bounded, u_t is a quasi-martingale, and the sum converges almost surely, thanks to the following quasi-martingale theorem:

Proposition 18 (Quasi-martingale Theorem [125, Section 4.4])
Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $\{u_t\}_{t>0}$ be a stochastic process on the probability space and $\{\mathcal{F}_t\}_{t>0}$ be a filtration by the past information at time instant t. Let us define the indicator function δ_t as follows

$$\delta_t \stackrel{\Delta}{=} \begin{cases} 1 & \text{if } \mathbb{E}[u_{t+1} - u_t | \mathcal{F}_t] > 0, \\ 0 & \text{otherwise.} \end{cases}$$

For all t, if $u_t \geq 0$ and $\sum_{i=1}^{\infty} \mathbb{E}[\delta_i(u_{i+1} - u_i)|\mathcal{F}_i] < \infty$, then u_t is a quasi-martingale and converges almost surely, i.e.,

$$\sum_{t=1}^{\infty} \mathbb{E}[u_{t+1} - u_t | \mathcal{F}_t] < \infty.$$

Now, we begin with the following relation when $L_t = t$

$$\tilde{f}_{t+1}(\mathbf{D}_t) = \frac{1}{t+1} \sum_{\tau=1}^{t+1} \beta^{t+1-k} \tilde{\ell}(\mathbf{D}_t, \boldsymbol{\mathcal{P}}_\tau, \boldsymbol{\mathcal{Y}}_\tau, \boldsymbol{\mathcal{O}}_\tau, \mathbf{u}_\tau)$$

$$= \frac{\tilde{\ell}(\mathbf{D}_t, \boldsymbol{\mathcal{P}}_{t+1}, \boldsymbol{\mathcal{Y}}_{t+1}, \boldsymbol{\mathcal{O}}_{t+1}, \mathbf{u}_{t+1})}{t+1} + \frac{t(\beta-1)}{t+1} \tilde{f}_t(\mathbf{D}_t) + \frac{t}{t+1} \tilde{f}_t(\mathbf{D}_t). \quad (6.173)$$

Thanks to Lemma 1 and $\lambda \leq 1$, we obtain $\tilde{f}_{t+1}(\mathbf{D}_{t+1}) \leq \tilde{f}_{t+1}(\mathbf{D}_t)$ and

$$\frac{\tilde{f}_t(\mathbf{D}_t) - f_t(\mathbf{D}_t)}{t+1} \leq \tilde{f}_t(\mathbf{D}_t) - \tilde{f}_{t+1}(\mathbf{D}_{t+1}) + \frac{\tilde{\ell}(\mathbf{D}_t, \boldsymbol{\mathcal{P}}_{t+1}, \boldsymbol{\mathcal{Y}}_{t+1}, \boldsymbol{\mathcal{O}}_{t+1}, \mathbf{u}_{t+1}) - f_t(\mathbf{D}_t)}{t+1}.$$

$$(6.174)$$

Since $f_t(\mathbf{D}_t) \leq \tilde{f}_t(\mathbf{D}_t) \ \forall t$, we have

$$\tilde{f}_{t+1}(\mathbf{D}_{t+1}) - \tilde{f}_t(\mathbf{D}_t) \leq \frac{\tilde{\ell}(\mathbf{D}_t, \boldsymbol{\mathcal{P}}_{t+1}, \boldsymbol{\mathcal{Y}}_{t+1}, \boldsymbol{\mathcal{O}}_{t+1}, \mathbf{u}_{t+1}) - f_t(\mathbf{D}_t)}{t+1}, \quad (6.175)$$

Define by $\{\mathcal{F}_t\}_{t>0}$ a filtration associated to $\{u_t\}_{t>0}$ where $\mathcal{F}_t = \{\mathbf{D}_k, \boldsymbol{O}_k, \mathbf{u}_k\}_{1 \leq k \leq t}$ records all past estimates of RACP at time t. By definition, for every $i \leq t$, $\mathcal{F}_i \subseteq \mathcal{F}_t$, and thus, the filtration is interpreted as streams of all historical but not future information generated by RACP. Now, taking the expectation of the inequality (D3) conditioned on $\mathcal{F}_t$ results in $\mathbb{E}\big[\tilde{f}_{t+1}(\mathbf{D}_{t+1}) - \tilde{f}_t(\mathbf{D}_t)|\mathcal{F}_t\big] \leq \dfrac{f(\mathbf{D}_t) - f_t(\mathbf{D}_t)}{t+1}$, where $\mathcal{F}_t$ is the filtration of past estimations at each time t; $f(.)$ is given by $f(\mathbf{D}) = \lim\limits_{k \to \infty} f_\tau(\mathbf{D})$, $\mathbb{E}\big[\ell(\mathbf{D}_t, \boldsymbol{P}_{k+1}\boldsymbol{X}_{k+1})\big] = f(\mathbf{D}_t), \forall \mathbf{D}_t$ and $\forall t$; and $\ell(\mathbf{D}_t, \boldsymbol{P}_{t+1}, \boldsymbol{Y}_{t+1}) = \tilde{\ell}(\mathbf{D}_t, \boldsymbol{P}_{t+1}, \boldsymbol{Y}_{t+1}, \boldsymbol{O}_{t+1}, \mathbf{u}_{t+1})$ due to $\{\boldsymbol{O}_{t+1}, \mathbf{u}_{t+1}\} = \arg\min_{\boldsymbol{O},\mathbf{u}} \tilde{\ell}(\mathbf{D}, \boldsymbol{P}_{t+1}, \boldsymbol{Y}_{t+1}, \boldsymbol{O}, \mathbf{u})$ at time t.

Next, let us define the following indicator function

$$\delta_t \triangleq \begin{cases} 1 & \text{if } \mathbb{E}\big[\tilde{f}_{t+1}(\mathbf{D}_{t+1}) - \tilde{f}_t(\mathbf{D}_t)|\mathcal{F}_t\big] > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6.176}$$

Here, the process $\{\delta_t\}_{t>0}$ is adapted to the filtration $\{\mathcal{F}_t\}_{t>0}$ as δ_t is measurable with respect to $\mathcal{F}_t$ for very t. From (D4), we then obtain Accordingly, we obtain $\mathbb{E}\Big[\delta_t \mathbb{E}\big[\tilde{f}_{t+1}(\mathbf{D}_{t+1}) - \tilde{f}_t(\mathbf{D}_t)|\mathcal{F}_t\big]\Big] \leq \mathbb{E}\Big[\sqrt{t}\big(f(\mathbf{D}_t) - f_t(\mathbf{D}_t)\big)\Big]\dfrac{1}{\sqrt{t}(t+1)}$. We know that the centered and scaled version of $f_t(\mathbf{D}_t)$ satisfies $\mathbb{E}\big[\sqrt{t}\big(f(\mathbf{D}_t) - f_t(\mathbf{D}_t)\big)\big] = \mathcal{O}(1)$, thanks to the Donsker theorem [126, Section 19.2]. We also derive $\displaystyle\int_{t=1}^{\infty} \dfrac{1}{\sqrt{t}(t+1)} dt < \infty$ after some simple calculations, thus $\displaystyle\sum_{t}^{\infty} \dfrac{1}{\sqrt{t}(t+1)} < \infty$ too. Accordingly, we obtain $\sum_{t=1}^{\infty} \mathbb{E}\big[\delta_t \mathbb{E}\big[\tilde{f}_{t+1}(\mathbf{D}_{t+1}) - \tilde{f}_t(\mathbf{D}_t)|\mathcal{F}_t\big]\big] < \infty$. Therefore, $\{\tilde{f}_t(\mathbf{D}_t)\}_{t=1}^{\infty}$ converges almost surely, i.e.,

$$\sum_{t=1}^{\infty} \mathbb{E}\big[\tilde{f}_{t+1}(\mathbf{D}_{t+1}) - \tilde{f}_t(\mathbf{D}_t)|\mathcal{F}_t\big] < \infty. \tag{6.177}$$

thanks to the quasi-martingale theorem [343, Theorem 9.4 & Proposition 9.5]

As $t \to \infty$, $\tilde{f}_t(\mathbf{D}_t) \to f_t(\mathbf{D}_t)$ almost surely

We prove $\{f_t(\mathbf{D}_t)\}_{t=1}^{\infty}$ and $\{\tilde{f}_t(\mathbf{D}_t)\}_{t=1}^{\infty}$ converge to the same limit by showing

$$\sum_{t=1}^{\infty} \frac{\tilde{f}_t(\mathbf{D}_t) - f_t(\mathbf{D}_t)}{t+1} < \infty. \tag{6.178}$$

According to (6.174), we have $e_t(\mathbf{D}_t)/t + 1$ is bounded by $\tilde{f}_t(\mathbf{D}_t) - \tilde{f}_{t+1}(\mathbf{D}_{t+1})$ and $\big(\ell(\mathbf{D}_t, \boldsymbol{P}_{t+1}, \boldsymbol{Y}_{t+1}) - f_t(\mathbf{D}_t)\big) / (t+1)$. Moreover, we have $\sum_{t=1}^{\infty} \tilde{f}_t(\mathbf{D}_t) - \tilde{f}_{t+1}(\mathbf{D}_{t+1}) < \infty$, and the sum of $\big(\ell(\mathbf{D}_t, \boldsymbol{P}_{t+1}, \boldsymbol{Y}_{t+1}) - f_t(\mathbf{D}_t)\big) / (t+1)$ also converges due to the convergence of $\mathbb{E}[f(\mathbf{D}_t) - f_t(\mathbf{D}_t)] / (t+1)$ and $\mathbb{E}[\ell(\mathbf{D}_t, \boldsymbol{P}, \boldsymbol{X})]$

$= f(\mathbf{D}_t) \forall t$. Since $\sum_{t=1}^{\infty} \frac{1}{t+1} = \infty$ and $\left| e_t(\mathbf{D}_t) - e_{t-1}(\mathbf{D}_{t-1}) \right| = O(1/t)$, we obtain $\sum_{t=1}^{\infty} \tilde{f}_t(\mathbf{D}_t) - f_t(\mathbf{D}_t) < \infty$, or

$$\tilde{f}_t(\mathbf{D}_t) \to f_t(\mathbf{D}_t) \ a.s., \tag{6.179}$$

thanks to [120, Lemma 3].

6.6.4   Appendix D: Proof of Lemma 13

In what follows, we prove that when $t \to \infty$, $\nabla \tilde{f}_t(\mathbf{D}_t) \to \nabla f_t(\mathbf{D}_t)$ and $\nabla \tilde{f}_t(\mathbf{D}_t) \to 0$ almost surely.

As $t \to \infty$, $\nabla \tilde{f}_t(\mathbf{D}_t) \to \nabla f_t(\mathbf{D}_t)$ almost surely

Let $\bar{\mathbf{D}} = \left[\bar{\mathbf{U}}^{(1)}, \bar{\mathbf{U}}^{(2)}, \dots, \bar{\mathbf{U}}^{(N)} \right]$ be the limit point of the sequence of solutions $\{\mathbf{U}_t^{(n)}\}_{t\geq 1}$.

We know that $\tilde{f}_t(\mathbf{D})$ is a majorant function of $f_t(\mathbf{D})$, i.e.,

$$\tilde{f}_t(\mathbf{D} + a_t \mathbf{V}) \geq f_t(\mathbf{D} + a_t \mathbf{V}) \ \forall \mathbf{D}, \mathbf{V} \in \mathcal{D}, a_t. \tag{6.180}$$

Taking the Taylor expansion of (6.180) at $t \to \infty$ results in

$$f_\infty(\bar{\mathbf{D}}) + \mathrm{tr}\left[a_t \mathbf{V}^\top \nabla f_\infty(\bar{\mathbf{D}}) \right] + o(a_t \mathbf{V}) \leq \tilde{f}_\infty(\bar{\mathbf{D}}) + \mathrm{tr}\left[a_t \mathbf{V}^\top \nabla \tilde{f}_\infty(\bar{\mathbf{D}}) \right] + o(a_t \mathbf{V}), \tag{6.181}$$

where $\tilde{f}_\infty = \lim_{t\to\infty} \tilde{f}_t(.)$. As indicated in Lemma 1, $\tilde{f}_\infty(\bar{\mathbf{D}}) = f_\infty(\bar{\mathbf{D}})$ and hence $\mathrm{tr}\left[a_t \mathbf{V}^\top \nabla f_\infty(\bar{\mathbf{D}}) \right] \leq \mathrm{tr}\left[a_t \mathbf{V}^\top \nabla \tilde{f}_\infty(\bar{\mathbf{D}}) \right]$. Since the above inequality must hold for all $\mathbf{V}$ and a_t, we obtain $\mathrm{tr}\left[\nabla \tilde{f}_\infty(\bar{\mathbf{D}}) - \nabla f_\infty(\bar{\mathbf{D}}) \right] \to 0 \ a.s.$ or

$$\nabla \tilde{f}_\infty(\bar{\mathbf{D}}) = \nabla f_\infty(\bar{\mathbf{D}}) \ \text{ almost surely}. \tag{6.182}$$

As $t \to \infty$, $\nabla \tilde{f}_\infty(\bar{\mathbf{D}}) = 0$

This property is proved by applying immediately the following stages:

1. Stage 1: $\lim_{t\to\infty} \mathrm{tr}\left[(\mathbf{D}_t - \mathbf{D}_{t+1})^\top \nabla \tilde{f}_{t+1}(\mathbf{D}_{t+1}) \right] = 0$;

2. Stage 2: $\mathrm{tr}\left[(\mathbf{D}_t - \mathbf{D}_{t+1})^\top \nabla \tilde{f}_{t+1}(\mathbf{D}_{t+1}) \right] \leq c_1 \mathrm{tr}\left[(\mathbf{D} - \mathbf{D}_t)^\top \nabla \tilde{f}_{t+1}(\mathbf{D}_t) \right] + c_2 \left\| \mathbf{D}_{t+1} - \mathbf{D}_t \right\|_F^2 \ \forall t, \mathbf{D} \in \mathcal{D}$;

3. Stage 3: $\left(\nabla \tilde{f}_t(\bar{\mathbf{D}}) \right)^\top (\mathbf{D} - \bar{\mathbf{D}}) \geq 0 \ \forall \mathbf{D}$ where $\bar{\mathbf{D}}$ is the limited point of the sequence $\{\mathbf{D}_t\}_{t\geq 1}$.

Stage 1:

When $L_t = t$, we can recast the surrogate function $\tilde{f}_t(.)$ into the following form

$$\tilde{f}_t(\mathbf{D}) = \frac{\rho}{t} \operatorname{tr} \left[\mathbf{A}_t \left(\left[(\mathbf{U}^{(N)})^\top \mathbf{U}^{(N)} \right] \circledast \left[(\mathbf{U}^{(N-1)})^\top \mathbf{U}^{(N-1)} \right] \circledast \cdots \circledast \left[(\mathbf{U}^{(1)})^\top \mathbf{U}^{(1)} \right] \right) \right]$$

$$- \frac{2\rho}{t} \operatorname{tr} \left[\mathbf{B}_t \left(\mathbf{U}^{(N)} \odot \mathbf{U}^{(N-1)} \odot \cdots \odot \mathbf{U}^{(1)} \right)^\top \right] + \mathbf{R}_{\mathcal{X},\mathcal{O}}, \quad (6.183)$$

where $\mathbf{A}_t = \lambda \mathbf{A}_{t-1} + \mathbf{u}_t \mathbf{u}_t^\top$, and $\mathbf{B}_t$ is the $(N+1)$-unfolding matrix of the tensor $\boldsymbol{\mathcal{B}}_t = \lambda \boldsymbol{\mathcal{B}}_{t-1} + \boldsymbol{\mathcal{P}}_t \circledast (\boldsymbol{\mathcal{Y}}_t - \boldsymbol{\mathcal{O}}_t) \times_{N+1} \mathbf{u}_t^\top$, and $\mathbf{R}_{\mathcal{X},\mathcal{O}} = \frac{\rho}{t} \sum_{\tau=1}^t \|\boldsymbol{\mathcal{P}}_t \circledast \boldsymbol{\mathcal{Y}}_t\|_F^2 + \frac{1}{t} \sum_{\tau=1}^t \beta^{t-\tau} \|\boldsymbol{\mathcal{O}}_\tau\|_1$ independent of $\mathbf{D}$. With respect to each factor $\mathbf{U}^{(n)}$, we can further express $\tilde{f}_t(\mathbf{D})$ as follows

$$\tilde{f}_t(\mathbf{D}) = \frac{\rho}{t} \operatorname{tr} \left[(\mathbf{U}^{(n)})^\top \mathbf{U}^{(n)} \mathbf{A}_{t,n} \right] - \frac{2\rho}{t} \operatorname{tr} \left[(\mathbf{U}^{(n)})^\top \mathbf{B}_{t,n} \right] + \mathbf{R}_{\mathcal{X},\mathcal{O}}. \quad (6.184)$$

Here, the two matrices $\mathbf{A}_{t,n}$ and $\mathbf{B}_{t,n}$ are given by

$$\mathbf{A}_{t,n} = \mathbf{A}_t \circledast \left[(\mathbf{U}^{(1)})^\top \mathbf{U}^{(1)} \right] \circledast \cdots \circledast \left[(\mathbf{U}^{(n-1)})^\top \mathbf{U}^{(n-1)} \right] \quad (6.185)$$

$$\circledast \left[(\mathbf{U}^{(n+1)})^\top \mathbf{U}^{(n+1)} \right] \circledast \cdots \circledast \left[(\mathbf{U}^{(1)})^\top \mathbf{U}^{(1)} \right],$$

$$\mathbf{B}_{t,n} = \sum_{j=1}^r \mathbf{B}_t^{(j)} \times_1 \mathbf{U}^{(1)}(:, j) \times_2 \cdots \times_{n-1} \mathbf{U}^{(n-1)}(:, j) \quad (6.186)$$

$$\times_{n+1} \mathbf{U}^{(n+1)}(:, j) \cdots \times_N \mathbf{U}^{(N)}(:, j),$$

where $\mathbf{B}_t^{(j)} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ denote the j-th mode-$(N + 1)$ slices of $\mathbf{B}_t$. It is easy to see that $\tilde{f}_t(\mathbf{D})$ is a multi-block convex and differentiable function and its partial derivative w.r.t. each block is Lipschitz continuous with constant $\tilde{L}_{t,n} = \|\mathbf{A}_{t,n}\|_F$. Accordingly, we have

$$\left| \tilde{f}_{t+1}(\mathbf{D}_t) - \tilde{f}_{t+1}(\mathbf{D}_{t+1}) - \operatorname{tr} \left[(\mathbf{D}_t - \mathbf{D}_{t+1})^\top \nabla \tilde{f}_{t+1}(\mathbf{D}_{t+1}) \right] \right| \le \tilde{L} \|\mathbf{D}_t - \mathbf{D}_{t+1}\|_F^2, \quad (6.187)$$

with $\tilde{L} = \max_n (\tilde{L}_{t,n}/2)$. Thanks to the triangle inequality, we then obtain

$$\left| \operatorname{tr} \left[(\mathbf{D}_t - \mathbf{D}_{t+1})^\top \nabla \tilde{f}_{t+1}(\mathbf{D}_{t+1}) \right] \right| \le \tilde{L} \|\mathbf{D}_t - \mathbf{D}_{t+1}\|_F^2 + \tilde{f}_{t+1}(\mathbf{D}_t) - \tilde{f}_{t+1}(\mathbf{D}_{t+1}). \quad (6.188)$$

Accordingly, we have

$$\sum_{t=1}^\infty \left| \mathbb{E} \left[\operatorname{tr} \left[(\mathbf{D}_t - \mathbf{D}_{t+1})^\top \nabla \tilde{f}_{t+1}(\mathbf{D}_{t+1}) \right] \big| \mathcal{F}_t \right] \right|$$

$$\le \tilde{L} \sum_{t=1}^\infty \mathbb{E} \left[\|\mathbf{D}_t - \mathbf{D}_{t+1}\|_F^2 \right] + \sum_{t=1}^\infty \left| \mathbb{E} \left[\tilde{f}_{t+1}(\mathbf{D}_{t+1}) - \tilde{f}_{t+1}(\mathbf{D}_t) \big| \mathcal{F}_t \right] \right|. \quad (6.189)$$

Recall that $\|\mathbf{D}_t - \mathbf{D}_{t+1}\|_F = O(1/t)$ as indicated in Proposition 1, hence $\sum_{t=1}^{\infty}$ $\|\mathbf{D}_t - \mathbf{D}_{t+1}\|_F^2 \le d \sum_{t=1}^{\infty} \frac{1}{t^2} = d\frac{\pi}{6} < \infty$ for some constant $d > 0$. Together with (6.177), we obtain that the right hand side of (6.189) is finite.

Also, it is well-known that $\mathbb{E}[|x|] < \infty$ implies $|x| < \infty$ almost surely for any random variable x, thus we obtain

$$\sum_{t=1}^{\infty} \left| \operatorname{tr}\left[(\mathbf{D}_t - \mathbf{D}_{t+1})^{\top} \nabla \tilde{f}_{t+1}(\mathbf{D}_{t+1}) \right] \right| < \infty. \tag{6.190}$$

Moreover, we always have

$$\sum_{t=1}^{\infty} \operatorname{tr}\left[(\mathbf{D}_t - \mathbf{D}_{t+1})^{\top} \nabla \tilde{f}_{t+1}(\mathbf{D}_{t+1}) \right] < \sum_{t=1}^{\infty} \left| \operatorname{tr}\left[(\mathbf{D}_t - \mathbf{D}_{t+1})^{\top} \nabla \tilde{f}_{t+1}(\mathbf{D}_{t+1}) \right] \right| < \infty. \tag{6.191}$$

Therefore the series $\left\{ \operatorname{tr}[(\mathbf{D}_t - \mathbf{D}_{t+1})^{\top} \nabla \tilde{f}_{t+1}(\mathbf{D}_{t+1})] \right\}_{t \ge 1}$ converges and we suppose that it converges to $C < \infty$.

Now, we rewrite (6.191) as follows

$$\lim_{t \to \infty} \sum_{\tau=1}^{t} \operatorname{tr}\left[(\mathbf{D}_k - \mathbf{D}_{k+1})^{\top} \nabla \tilde{f}_{k+1}(\mathbf{D}_{k+1}) \right] = \lim_{t \to \infty} \operatorname{tr}\left[(\mathbf{D}_t - \mathbf{D}_{t+1})^{\top} \nabla \tilde{f}_{t+1}(\mathbf{D}_{t+1}) \right]$$

$$+ \lim_{t \to \infty} \sum_{\tau=1}^{t-1} \operatorname{tr}\left[(\mathbf{D}_k - \mathbf{D}_{k+1})^{\top} \nabla \tilde{f}_{k+1}(\mathbf{D}_{k+1}) \right] = C < \infty. \tag{6.192}$$

When $t \to \infty$, the following partial sum also converges to C, i.e.,

$$\lim_{t \to \infty} \sum_{\tau=1}^{t-1} \operatorname{tr}\left[(\mathbf{D}_k - \mathbf{D}_{k+1})^{\top} \nabla \tilde{f}_{k+1}(\mathbf{D}_{k+1}) \right] = C. \tag{6.193}$$

It implies that

$$\lim_{t \to \infty} \operatorname{tr}\left[(\mathbf{D}_t - \mathbf{D}_{t+1})^{\top} \nabla \tilde{f}_{t+1}(\mathbf{D}_{t+1}) \right] = 0. \tag{6.194}$$

Step 2:

Because $\mathbf{U}_{t+1}^{(n)} = \operatorname{argmin}_{\mathbf{U}^{(n)}} \tilde{f}_{t+1}(\mathbf{U}^{(n)}, .)$, we have

$$\tilde{f}_{t+1}\left(\mathbf{U}_{t+1}^{(n)}, .\right) \le \tilde{f}_{t+1}\left(\mathbf{U}_t^{(n)} + \frac{d_1}{tN}(\mathbf{U}^{(n)} - \mathbf{U}_t^{(n)}), .\right) \; \forall \mathbf{D} \in \mathcal{D}. \tag{6.195}$$

Without loss of generality, we suppose that $\mathbf{D}$ is arbitrarily chosen in $\mathcal{D}$ such that $\|\mathbf{D} - \mathbf{D}_t\|_F = d_1/tN$ for some positive constant $d_1 > 0$, hence $\|\mathbf{U}^{(n)} - \mathbf{U}_t^{(n)}\|_F \le d_1/Nt \; \forall n$.

As mentioned in Stage 1, $\nabla \tilde{f} = \left[\nabla_1 \tilde{f}, \nabla_2 \tilde{f}, \ldots, \nabla_N \tilde{f}\right]$ is Lipschitz where $\nabla_n \tilde{f}$ denote the partial derivative of $\tilde{f}$ w.r.t. the n-th factor $\mathbf{U}^{(n)}$. Thanks to Proposition 22, there always exists a constant $d_2 > 0$ such that

$$\text{tr}\left[\left(\mathbf{U}_t^{(n)} - \mathbf{U}_{t+1}^{(n)}\right)^\top \nabla_n \tilde{f}_{t+1}\left(\mathbf{U}_{t+1}^{(n)}, .\right)\right]$$

$$\leq \frac{d_1}{tN} \text{tr}\left[\left(\mathbf{U}^{(n)} - \mathbf{U}_t^{(n)}\right)^\top \nabla_n \tilde{f}_{t+1}\left(\mathbf{U}_t^{(n)}, .\right)\right] + \frac{\tilde{L}d_2}{t^2N^2}. \tag{6.196}$$

Collecting these inequalities with $n = 1, 2, \ldots, N$ together, we derive

$$\text{tr}\left[\left(\mathbf{D}_t - \mathbf{D}_{t+1}\right)^\top \left[\nabla_1 \tilde{f}_{t+1}\left(\mathbf{U}_{t+1}^{(1)}, .\right), \nabla_2 \tilde{f}_{t+1}\left(\mathbf{U}_{t+1}^{(2)}, .\right), \ldots, \nabla_N \tilde{f}_{t+1}\left(\mathbf{U}_{t+1}^{(N)}, .\right)\right]\right]$$

$$\leq \frac{d_1}{tN} \text{tr}\left[\left(\mathbf{D} - \mathbf{D}_t\right)^\top \left[\nabla_1 \tilde{f}_{t+1}\left(\mathbf{U}_t^{(n)}, .\right), \nabla_2 \tilde{f}_{t+1}\left(\mathbf{U}_t^{(n)}, .\right), \ldots, \nabla_N \tilde{f}_{t+1}\left(\mathbf{U}_t^{(n)}, .\right)\right]\right]$$

$$+ \frac{\tilde{L}d_2}{t^2N^2}. \tag{6.197}$$

It then follows that

$$\text{tr}\left[\left(\mathbf{D}_t - \mathbf{D}_{t+1}\right)^\top \nabla \tilde{f}_{t+1}\left(\mathbf{D}_{t+1}\right)\right] \leq \frac{d_1}{tN} \text{tr}\left[\left(\mathbf{D} - \mathbf{D}_t\right)^\top \nabla \tilde{f}_{t+1}\left(\mathbf{D}_t\right)\right]$$

$$+ \tilde{L}d_2 \left\|\mathbf{D}_t - \mathbf{D}_{t+1}\right\|_F^2, \tag{6.198}$$

because of $\left\|\mathbf{D}_t - \mathbf{D}_{t+1}\right\|_F = O(1/t)$. The inequality (6.198) still holds for all $\mathbf{D} \in \mathcal{D}$ such that $\left\|\mathbf{D} - \mathbf{D}_t\right\|_F > d_1/tN$.

Step 3:

We use the proof by contradiction to indicate that the limited point $\bar{\mathbf{D}}$ is a stationary point of $\tilde{f}_\infty(.)$ over $\mathcal{D}$.

Assume that $\bar{\mathbf{D}}$ is not a stationary point of $\tilde{f}_t$ over $\mathbf{D}$ when $t \to \infty$. Then there exists $\mathbf{D}' \in \mathcal{D}$ and $\epsilon_1 > 0$ such that

$$\text{tr}\left[\left(\mathbf{D}' - \bar{\mathbf{D}}\right)^\top \nabla \tilde{f}_\infty(\bar{\mathbf{D}})\right] \leq -\epsilon_1 < 0. \tag{6.199}$$

Thanks to the triangle inequality, we have

$$\left\|\left(\mathbf{D}' - \mathbf{D}_\tau\right)^\top \nabla \tilde{f}_{k+1}\left(\mathbf{D}_\tau\right) - \left(\mathbf{D}' - \bar{\mathbf{D}}\right)^\top \nabla \tilde{f}_\infty(\bar{\mathbf{D}})\right\|_F$$

$$\leq \left\|\nabla \tilde{f}_{k+1}\left(\mathbf{D}_\tau\right) - \nabla \tilde{f}_\infty(\bar{\mathbf{D}})\right\|_F \left\|\mathbf{D}' - \mathbf{D}_\tau\right\|_F + \left\|\nabla \tilde{f}_\infty(\bar{\mathbf{D}})\right\|_F \left\|\bar{\mathbf{D}} - \mathbf{D}_\tau\right\|_F. \tag{6.200}$$

It is easy to see that the RHS of (6.200) approaches to zero as $k \to \infty$ because of $\mathbf{D}_\tau \to \bar{\mathbf{D}}$ and $\nabla \tilde{f}_{k+1}\left(\mathbf{D}_\tau\right) \to \nabla \tilde{f}_\infty(\bar{\mathbf{D}})$. In parallel, we know that $\text{tr}[\mathbf{A}] - \text{tr}[\mathbf{B}] = \text{tr}[\mathbf{A} - \mathbf{B}] \leq \sqrt{n}\|\mathbf{A} - \mathbf{B}\|_F$ and hence

$$\text{tr}\left[\left(\mathbf{D}' - \mathbf{D}_k\right)^\top \nabla \tilde{f}_{k+1}\left(\mathbf{D}_k\right)\right] \leq -\epsilon_1 < 0. \tag{6.201}$$

According to (6.198), we obtain

$$\lim_{k \to \infty} \text{tr}\left[(\mathbf{D}_\tau - \mathbf{D}_{k+1})^\top \nabla \tilde{f}_{k+1}(\mathbf{D}_{k+1}) \right] \leq \frac{-d_1 \epsilon}{tN\|\mathbf{D}' - \mathbf{D}_\tau\|_F} < 0, \qquad (6.202)$$

which is a contradiction in (6.194) in Step 1. Therefore, $\bar{\mathbf{D}}$ is a stationary point of $\tilde{f}_\infty$.

6.6.5   Appendix E: Useful Propositions

In this section, we would provide the following propositions which help us to derive several important results in the proofs. Their details are provided in well-known materials.

Proposition 19 ([132, Section 9.1.2]) *The function f is m-strongly convex, with a constant m if and only if for all $\mathbf{u}, \mathbf{v} \in \mathbf{dom}(f)$, we always have $|f(\mathbf{v}) - f(\mathbf{u})| \geq \frac{m}{2} \|\mathbf{v} - \mathbf{u}\|^2$.*

Proposition 20 ([132, page 72]) *Every norm on $\mathbb{R}^n$ is convex and the sum of convex functions is convex.*

Proposition 21 ([132, page 329]) *A function $f : \mathcal{V} \to \mathbb{R}$ is called Lipschitz function if there exist a positive number $L > 0$ such that for all $\mathbf{A}, \mathbf{B} \in \mathcal{V}$, we always have $|f(\mathbf{A}) - f(\mathbf{B})| \leq L\|\mathbf{A} - \mathbf{B}\|$.*

Proposition 22 ([353, Lemma 1.2.3]) *If a function $f : \mathcal{V} \to \mathbb{R}$ is differentiable and its derivative is L-Lipschitz continuous, then for all $\mathbf{A}, \mathbf{B} \in \mathcal{V}$,*

$$\left| f(\mathbf{A}) - f(\mathbf{B}) - \left(\nabla f(\mathbf{B})\right)^\top (\mathbf{A} - \mathbf{B}) \right| \leq \frac{L}{2} \|\mathbf{A} - \mathbf{B}\|^2.$$

Proposition 23 *If $\{f_t\}_{t \geq 1}$ and $\{g_t\}_{t \geq 1}$ are sequences of bounded functions which converge uniformly on a set $\mathcal{E}$, then $\{f_t + g_t\}_{t \geq 1}$ and $\{f_t g_t\}_{t \geq 1}$ converge uniformly on $\mathcal{E}$.*

Proof. Since f_t and g_t are bounded, we obtain $|f_t| < M < \infty$ and $|g_t| < N < \infty$ for all t. The triangle inequality gives $|f_t + g_t| \leq |f_t| + |g_t| < M + N$ for all t. Also, $|f_t g_t| = |f_t||g_t| \leq MN$. Therefore $f_t + g_t$ and $f_t g_t$ are bounded.

Proposition 24 ([120, Lemma 3, page 35]) *Let $\{a_t\}_{t=1}^\infty$ and $\{b_t\}_{t=1}^\infty$ be two nonnegative sequences such that $\sum_{i=1}^\infty a_i = \infty$ and $\sum_{i=1}^\infty a_i b_i < \infty$, $|b_{t+1} - b_t| < K a_t$ with some constant K, then $\lim_{t \to \infty} b_t = 0$ or $\sum_{i=1}^\infty b_i < \infty$.*

Proposition 25 ([350, Theorem 4.1, page 237]) *Consider a continuous function $f : \mathcal{V} \times \mathcal{U} \to \mathbb{R}$. Suppose that $\forall \mathbf{u} \in \mathcal{U}$, the function $f(., \mathbf{u})$ is differentiable and $\nabla_{\mathbf{v}} f(\mathbf{v}, \mathbf{u})$ is continuous on $\mathcal{V} \times \mathcal{U}$. If $g(\mathbf{v})$ be the function derived from $g(\mathbf{v}) = \min_{\mathbf{u} \in \mathcal{U}} f(\mathbf{v}, \mathbf{u})$, then $g(\mathbf{v})$ is also differentiable. In addition, if $\mathbf{u}^* = \operatorname{argmin}_{\mathbf{u} \in \mathcal{U}} f(\mathbf{v}, \mathbf{u})$ be unique, $\nabla g(\mathbf{v}) = \nabla_{\mathbf{v}} f(\mathbf{v}, \mathbf{u}^*)$, $\forall \mathbf{v} \in \mathcal{V}$.*

Proposition 26 ($\mathbb{P}$-Donsker classes, Donsker theorem [126, Section 19.2]) *Let $F = \{\ell_\theta : \mathcal{X} \to \mathbb{R}\}$ be a set of measurable functions defined on a bounded subset of $\mathbb{R}^n$. For every θ_1, θ_2 and x, if there exists a constant c such that $|\ell_{\theta_1}(x) - \ell_{\theta_2}(x)| < c \, \|\theta_1 - \theta_2\|_2$, then F is $\mathbb{P}$-Donsker. For any function ℓ in F, let us define the following functions*

$$f_t = \frac{1}{t} \sum_{i=1}^{t} \ell(\mathbf{U}_i), \quad and \quad f = \mathbb{E}\big[f_t(\mathbf{U})\big].$$

Assume that for all ℓ, $\|\ell\|_\infty < M$ and random variables $\{\mathbf{U}_i\}_{i \geq 1}$ are Borel-measurable, we then have $\mathbb{E}[\sqrt{t}\|f_t - f\|_\infty] = O(1)$, where $\|\ell\|_\infty \overset{\Delta}{=} \inf\{C \geq 0, |f(x)| < C \ \forall \ x\}$.

Proposition 27 (Quasi Martingales [125, Section 4.4]) *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $\{u_t\}_{t>0}$ be a stochastic process on the probability space and $\{\mathcal{F}_t\}_{t>0}$ be a filtration by the past information at time instant t. Let us define the indicator function δ_t as follows*

$$\delta_t \overset{\Delta}{=} \begin{cases} 1 & if \ \ \mathbb{E}[u_{t+1} - u_t | \mathcal{F}_t] > 0, \\ 0 & otherwise. \end{cases}$$

For all t, if $u_t \geq 0$ and $\sum_{i=1}^{\infty} \mathbb{E}[\delta_i(u_{i+1} - u_i)|\mathcal{F}_i] < \infty$, then u_t is a quasi-martingale and converges almost surely, i.e.,

$$\sum_{t=1}^{\infty} \mathbb{E}[u_{t+1} - u_t | \mathcal{F}_t] < \infty.$$

Tensor Tracking under Tensor-Train Format

7

Tensor-train (TT) decomposition has been an efficient tool to find low order approximation of large-scale, high-order tensors. In online setting, TT decomposition has not gained much attention and popularity as CP and Tucker decompositions. In particular, the existing TT decomposition algorithms are either of high computational complexity or operating in batch-mode, and hence, they become inefficient for (near) real-time processing. In this chapter, we introduce three new online algorithms for the problem of streaming tensor-train decomposition. The first algorithm called TT-FOA is capable of tracking the low-rank components of high-order tensors from noisy and high-dimensional data with high accuracy, even when they come from time-dependent observations. The second algorithm called ATT is specifically designed for handling incomplete streaming tensors. ATT is scalable, effective, and adept at estimating low TT-rank component of streaming tensors. To deal with sparse outliers, we propose the so-called ROBOT algorithm which stands for ROBust Online Tensor-Train decomposition. Technically, ROBOT has the ability to tracking streaming tensors from imperfect streams (i.e., due to noise, outliers, and missing data) as well as tracking their time variation in dynamic environments. We conduct several experiments on both synthetic and real data to demonstrate the effectiveness of the proposed algorithms.

7.1   Introduction

Tensor decomposition has received increasing attention from the machine learning and signal processing community over the years [10, 11]. It has been successfully applied to a broad range of applications, from wireless communications [182, 354] and image processing [355, 356] to neuroscience [179, 357]. 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 [12]. Under the tensor-train format, we can factorize a high-order tensor into a sequence of 3-order tensors, see Fig. 7.1 for an illustration.
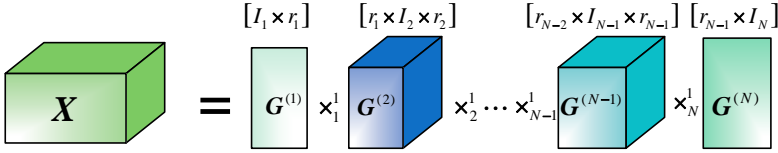
Figure 7.1: Tensor-train decomposition of $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$.

TT decomposition offers several advantages compared to the two standard Tucker and CP/PARAFAC decompositions. First, we can represent any high-order tensor under TT decomposition and its computation is stable since it is based on computing low-rank approximations of unfolding matrices of the tensor [16]. Second, TT-rank can be effectively determined in a stable way in contrast to CP-rank which is known as an NP-hard problem [195, 358]. 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 [16, 189]. Accordingly, TT decomposition is expected to be capable of handling big tensors efficiently and effectively. We refer the readers to [12] for a comprehensive survey on basic properties, algorithms, and applications of the tensor-train decomposition.

In recent years, the demand for big data stream analysis has been increasing rapidly [2]. 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 [359]. Accordingly, it is of great interest to develop adaptive (online) 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. 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 [360–362], Lubich *et al.* introduced some dynamical tensor approximation methods under TT format for factorizing time-varying tensors, thanks to the Dirac–Frenkel-McLachlan vari-
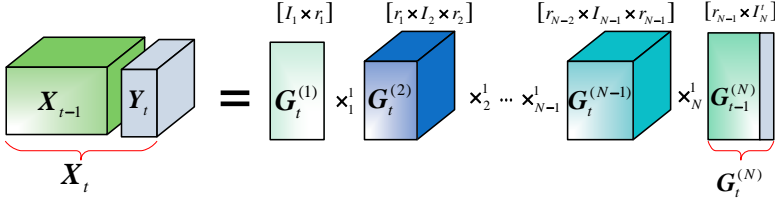
Figure 7.2: Streaming Tensor-Train Decomposition of $\boldsymbol{\mathcal{X}}_t \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times I_N^t}$.

ational principle. However, the dynamical tensors of interest are of fixed size, and hence, their methods indeed belong to the class of batch TT algorithms. In [267], 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 [268], 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. These drawbacks encourage us to develop adaptive methods for factorizing high-order streaming tensors under the tensor-train format.

7.2 Streaming Tensor-Train Decomposition

7.2.1 Problem Formulation

Consider a streaming N-order 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" dimension I_N^t. At time t, $\boldsymbol{\mathcal{X}}_t$ is particularly obtained by appending a new slice $\boldsymbol{\mathcal{Y}}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1}}$ to the previous observation $\boldsymbol{\mathcal{X}}_{t-1}$ along the time dimension, i.e., $I_N^t = I_N^{t-1} + 1$, please Fig. 7.2 for an illustration. Instead of recomputing the batch TT decomposition for $\boldsymbol{\mathcal{X}}_t$, we aim to develop an efficient update, both in computational complexity and memory storage, to obtain TT-cores of $\boldsymbol{\mathcal{X}}_t$ from past estimations.

TT decomposition of $\boldsymbol{\mathcal{X}}_t$ can be represented by a multilinear product of 3-order tensors called TT-cores:

$$\boldsymbol{\mathcal{X}}_t = \boldsymbol{\mathcal{G}}_t^{(1)} \times_2^1 \boldsymbol{\mathcal{G}}_t^{(2)} \times_3^1 \cdots \times_N^1 \boldsymbol{\mathcal{G}}_t^{(N)}, \tag{7.1}$$

where $\mathbf{r}_{\mathrm{TT}} = [r_1, r_2, \ldots, r_{N-1}]$ is a vector containing the TT-ranks, $\boldsymbol{\mathcal{G}}_t^{(1)} \in$

$\mathbb{R}^{I_1 \times r_1}$, $\boldsymbol{G}_t^{(N)} \in \mathbb{R}^{r_{N-1} \times I_N^t}$ and $\boldsymbol{G}_t^{(n)} \in \mathbb{R}^{r_{n-1} \times I_n \times r_n}$, $n = 2, \ldots, N-1$, are the TT-cores. In practice, (7.1) is only an approximate model in a noisy environment, i.e.,

$$\boldsymbol{X}_t = \boldsymbol{G}_t^{(1)} \times_2^1 \boldsymbol{G}_t^{(2)} \times_3^1 \cdots \times_N^1 \boldsymbol{G}_t^{(N)} + \boldsymbol{N}_t \tag{7.2}$$

where $\boldsymbol{N}_t$ is a noise tensor. The TT-cores can be estimated by solving the following minimization:

$$\{\boldsymbol{G}_t^{(n)}\}_{n=1}^N = \underset{\{\boldsymbol{G}^{(n)}\}_{n=1}^N}{\operatorname{argmin}} \frac{1}{2} \left\| \boldsymbol{X}_t - \widetilde{\boldsymbol{X}} \right\|_F^2 \quad \text{s.t.} \ \ \widetilde{\boldsymbol{X}} = \boldsymbol{G}^{(1)} \times_2^1 \boldsymbol{G}^{(2)} \times_3^1 \cdots \times_N^1 \boldsymbol{G}^{(N)}. \tag{7.3}$$

Problem (7.3) can be rewritten in the adaptive scheme as follows

$$\{\boldsymbol{G}_t^{(n)}\}_{n=1}^N = \underset{\{\boldsymbol{G}^{(n)}\}_{n=1}^N}{\operatorname{argmin}} \sum_{\tau=1}^t \beta^{t-\tau} \left\| \boldsymbol{Y}_\tau - \boldsymbol{G}^{(1)} \times_2^1 \cdots \times_{N-1}^1 \boldsymbol{G}^{(N-1)} \times_N^1 \mathrm{g}_\tau^{(N)} \right\|_F^2, \tag{7.4}$$

where $\boldsymbol{Y}_\tau \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1}}$ is the τ-th slice of $\boldsymbol{X}_t$, $\mathrm{g}_\tau^{(N)} \in \mathbb{R}^{r_{N-1} \times 1}$ is the i-th column of the last TT-core $\boldsymbol{G}_t^{(n)}$ and a forgetting factor $\lambda \in (0, 1]$ is to discount the effect of past observations. The following steps describe the basic idea of our method for solving (7.4).

Let us denote $\boldsymbol{H}_t = \boldsymbol{G}_t^{(1)} \times_2^1 \cdots \times_{N-1}^1 \boldsymbol{G}_t^{(N-1)}$, and $\{\boldsymbol{G}_{t-1}^{(n)}\}_{n=1}^N$ be the old estimated TT-cores of $\boldsymbol{X}_{t-1}$. Under the assumption that TT-cores are either static or changing slowly, hence $\boldsymbol{H}_t \simeq \boldsymbol{H}_{t-1}$. Thus, we have

$$\begin{aligned}
\boldsymbol{H}_t \times_N^1 \boldsymbol{G}_t^{(N)} &= \boldsymbol{X}_{t-1} \boxplus_N \boldsymbol{Y}_t \\
&= \left(\boldsymbol{H}_{t-1} \times_N^1 \boldsymbol{G}_{t-1}^{(N)} \right) \boxplus_N \left(\boldsymbol{H}_t \times_N^1 \mathrm{g}_t^{(N)} \right) \simeq \boldsymbol{H}_t \times_N^1 \left[\boldsymbol{G}_{t-1}^{(N)} \mid \mathrm{g}_t^{(N)} \right].
\end{aligned} \tag{7.5}$$

Accordingly, we only need to estimate the last column vector $\mathrm{g}_t^{(N)}$ of $\boldsymbol{G}_t^{(N)} \in \mathbb{R}^{r_{N-1} \times t}$ at time t, instead of re-estimating the whole $\boldsymbol{G}_t^{(N)}$ which becomes inefficient for a large t:

$$\boldsymbol{G}_t^{(N)} \simeq \left[\boldsymbol{G}_{t-1}^{(N)} \mid \mathrm{g}_t^{(N)} \right]. \tag{7.6}$$

The vector $\mathrm{g}_t^{(N)}$ can be updated by minimizing the t-th summand in (7.4):

$$\mathrm{g}_t^{(N)} = \underset{\mathrm{g}^{(N)} \in \mathbb{R}^{r_{N-1} \times 1}}{\operatorname{argmin}} \left\| \boldsymbol{Y}_t - \boldsymbol{H}_{t-1} \times_n^1 \mathrm{g}^{(N)} \right\|_F^2. \tag{7.7}$$

After that, we update TT-cores $\{\boldsymbol{G}^{(n)}\}_{n=1}^{N-1}$ by

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

(7.8)

where the two auxiliary tensors are given by

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

(7.9)

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

(7.10)

We make the following assumptions for convenience of deploying our method: (A1) TT-cores $\{\boldsymbol{G}^{(n)}\}_{n=1}^{N-1}$ may change slowly between two consecutive instances $t-1$ and t, i.e. $\boldsymbol{G}_t^{(n)} \simeq \boldsymbol{G}_{t-1}^{(n)}$; and (A2) TT-rank vector $\mathbf{r}_{\mathrm{TT}} = [r_1, r_2, \ldots, r_{N-1}]$ is known and does not change with time.

7.2.2 Proposed Method

In this subsection, we propose an efficient first-order method, namely TT-FOA (which stands for TT adaptive decomposition using First-Order Approach), for tensor-train decomposition of streaming tensors by adapting the alternating minimization framework to the problem (7.46). The proposed algorithm consists of two main steps: (i) estimate $\mathbf{g}_t^{(N)}$ first, given past estimated TT-cores; (ii) then we update TT-cores $\boldsymbol{G}^{(n)}$ in parallel, given $\mathbf{g}_t^{(N)}$ and remaining TT-cores. The pseudocode of TT-FOA is summarized in Algorithm 9.

7.2.2.1 Estimation of $\mathbf{g}_t^{(N)}$

Given a new slice $\boldsymbol{\mathcal{Y}}_t$ and past estimated TT-cores, $\mathbf{g}_t^{(N)}$ can be estimated by solving (7.7)

$$\mathbf{g}_t^{(N)} = \underset{\mathbf{g}^{(N)} \in \mathbb{R}^{r_{n-1} \times 1}}{\operatorname{argmin}} \left\| \boldsymbol{\mathcal{Y}}_t - \boldsymbol{\mathcal{H}}_{t-1} \times_N^1 \mathbf{g}^{(N)} \right\|_F^2 + \frac{\rho}{2} \left\| \mathbf{g}^{(N)} \right\|_2^2,$$

where ρ is a small positive parameter for regularization. It can be reformulated via its matrix-vector representation as follows

$$\mathbf{g}_t^{(N)} = \underset{\mathbf{g}^{(N)} \in \mathbb{R}^{r_{N-1} \times 1}}{\operatorname{argmin}} \left\| \mathbf{y}_t - \mathbf{H}_{t-1} \mathbf{g}^{(N)} \right\|_2^2 + \frac{\rho}{2} \left\| \mathbf{g}^{(N)} \right\|_2^2,$$

(7.11)

where $\mathbf{y}_t = \operatorname{vec}(\boldsymbol{\mathcal{Y}}_t)$ and $\mathbf{H}_{t-1} \in \mathbb{R}^{I_1 \ldots I_{N-1} \times r_{N-1}}$ is the unfolding matrix of $\boldsymbol{\mathcal{H}}_{t-1}$.

INPUT: Observations $\{\boldsymbol{\mathcal{Y}}_t\}_{t=1}^{\infty}$, $\boldsymbol{\mathcal{Y}}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1}}$, TT-rank $\mathbf{r}_{\text{TT}} = [r_1, r_2, \ldots, r_{N-1}]$, forgetting factor $0 < \beta \le 1$.

INITIALIZATION: $\{\boldsymbol{\mathcal{G}}_0^{(n)}\}_{n=1}^{N-1}$ are initialized randomly and $\{\mathbf{S}_0^{(n)}\}_{n=1}^{N-1} = \mathbf{I}$.

MAIN PROGRAM:

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

   Step 1: Estimate $\mathbf{g}_t^{(N)}$

   $\boldsymbol{\mathcal{H}}_{t-1} = \mathbf{G}_{t-1}^{(1)} \times_2^1 \mathbf{G}_{t-1}^{(2)} \times_3^1 \cdots \times_{N-1}^1 \boldsymbol{\mathcal{G}}_{t-1}^{(N-1)}$

   $\mathbf{H}_{t-1} = \texttt{unfolding}(\boldsymbol{\mathcal{H}}_{t-1}, [I_1 I_2 \ldots I_{N-1}, r_{N-1}])$

   $\Omega = \texttt{randsample}([1, I_1 I_2 \ldots I_{N-1}])$

   $\mathbf{y}_{\Omega_t} = \texttt{vec}(\boldsymbol{\mathcal{Y}}_t)$

   $\mathbf{g}_t^{(N)} = \mathbf{H}_{\Omega_{t-1}}^{\#} \mathbf{y}_{\Omega_t}$

   $\Delta_t = \boldsymbol{\mathcal{Y}}_t - \boldsymbol{\mathcal{H}}_{t-1} \times_N^1 \mathbf{g}_t^{(N)}$

   Step 2: Update TT-cores $\boldsymbol{\mathcal{G}}_k$ **in parallel**

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

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

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

   $\mathbf{B}_t^{(n)} = \texttt{unfolding}(\boldsymbol{\mathcal{B}}_t^{(n)}, [r_n, I_{n+1} I_{n+2} \ldots I_{N-1}])$

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

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

   $\mathbf{V}_t^{(n)} = \big(\mathbf{S}_t^{(n)}\big)^{-1} \mathbf{W}_t^{(n)\top}$

   $\Delta_t^{(n)} = \texttt{unfolding}(\Delta_t, [I_n, r_{n-1} r_n])$

   $\mathbf{G}_t^{(n)} = \mathbf{G}_{t-1}^{(n)} + \Delta_t^{(n)} \mathbf{V}_t^{(n)\top}$

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

end

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

Algorithm 9: TT-FOA: First-Order Adaptive Tensor-Train Decomposition

Problem (7.11) is an overdetermined least-squares (LS) regression, it can be efficiently solved by using the randomized sketching technique [323], as

$$\mathbf{g}_t^{(N)} = \operatorname*{argmin}_{\mathbf{g}^{(N)} \in \mathbb{R}^{r_{N-1} \times 1}} \left\| \mathcal{L}(\mathbf{H}_{t-1})\mathbf{g}^{(N)} - \mathcal{L}(\mathbf{y}_t) \right\|_2^2 + \frac{\rho}{2} \left\| \mathbf{g}^{(N)} \right\|_2^2, \qquad (7.12)$$

where $\mathcal{L}(.)$ is a sketching map. Thanks to the Kronecker structure of $\mathbf{H}[t-1]$, uniform random sampling can provide a good sketch for $\mathbf{H}_{t-1}$. Accordingly, we can select rows of $\mathbf{H}_{t-1}$ as well as $\mathbf{y}_t$ at random to form the sketch $\mathbf{H}_{\Omega_{t-1}} \in \mathbb{R}^{|\Omega| \times r_{N-1}}$ and a sampled vector $\in \mathbb{R}^{|\Omega| \times 1}$, where Ω denotes the set of sam-

pling rows. Therefore, $\mathbf{g}_t^{(N)}$ can be efficiently updated by applying the ridge regression method to (7.12), whose closed-form is given by

$$\mathbf{g}_t^{(N)} = \left(\mathbf{H}_{\Omega_{t-1}}^\top \mathbf{H}_{\Omega_{t-1}} + \rho \mathbf{I}_{r_{N-1}}\right)^{-1} \mathbf{H}_{\Omega_{t-1}}^\top \mathbf{y}_{\Omega_t}. \tag{7.13}$$

As a result, the last TT-core $\boldsymbol{\mathcal{G}}_t^{(N)}$ is updated as follows

$$\boldsymbol{\mathcal{G}}_t^{(N)} = \left[\boldsymbol{\mathcal{G}}_{t-1}^{(N)} \,\middle|\, \mathbf{g}_t^{(N)}\right]. \tag{7.14}$$

7.2.2.2 Estimation of TT-cores

Given the new slice $\boldsymbol{\mathcal{Y}}_t$ and past estimations, the k-th TT-core $\boldsymbol{\mathcal{G}}_t^{(n)}$ can be estimated by minimizing the matrix-representation of the objective function (7.8), as follows

$$\mathbf{G}_t^{(n)} = \underset{\mathbf{G}^{(n)} \in \mathbb{R}^{I_n \times r_n r_{n-1}}}{\operatorname{argmin}} \left[f(\mathbf{G}^{(n)}) = \sum_{\tau=1}^t \beta^{t-\tau} \left\| \mathbf{Y}_\tau^{(n)} - \mathbf{G}^{(n)} \mathbf{W}_\tau^{(n)} \right\|_F^2 \right], \tag{7.15}$$

where $\mathbf{G}_t^{(n)}$ is the mode-2 matricization of $\boldsymbol{\mathcal{G}}_t^{(n)}$, $\mathbf{Y}_\tau^{(n)}$ is the mode-n matricization of $\boldsymbol{\mathcal{Y}}_\tau$; $\mathbf{W}_\tau^{(n)} = \mathbf{B}_\tau^{(n)} \otimes \mathbf{A}_{t-1}^{(n)}$ where $\otimes$ denotes the Kronecker product, $\mathbf{A}_{t-1}^{(n)}$ and $\mathbf{B}_\tau^{(n)}$ are the unfolding matrices of $\boldsymbol{\mathcal{A}}_{t-1}^{(n)}$ and $\boldsymbol{\mathcal{B}}_\tau^{(n)}$ respectively;

The local optimal $\boldsymbol{\mathcal{G}}_t^{(n)}$ can be obtained by setting the first derivative of $f(\mathbf{G}^{(n)})$ to zero:

$$\mathbf{G}^{(n)} \sum_{i=1}^t \beta^{t-\tau} \mathbf{W}_\tau^{(n)} \mathbf{W}_\tau^{(n)\top} = \sum_{i=1}^t \beta^{t-\tau} \mathbf{Y}_\tau^{(n)} \mathbf{W}_\tau^{(n)\top}. \tag{7.16}$$

From that, we can obtain $\mathbf{G}_t^{(n)}$ in the recursive way as follows:

Let us denote $\mathbf{S}_t^{(n)} = \sum_{\tau=1}^t \beta^{t-\tau} \mathbf{W}_\tau^{(n)} \mathbf{W}_\tau^{(n)\top}$ and $\mathbf{R}_t^{(n)} = \sum_{\tau=1}^t \beta^{t-\tau} \mathbf{Y}_\tau^{(n)} \mathbf{W}_\tau^{(n)\top}$. The two matrices $\mathbf{R}_t^{(n)}$ and $\mathbf{S}_t^{(n)}$ can be updated recursively:

$$\mathbf{S}_t^{(n)} = \beta \mathbf{S}_{t-1}^{(n)} + \mathbf{W}_t^{(n)} \mathbf{W}_t^{(n)\top}, \tag{7.17}$$

$$\mathbf{R}_t^{(n)} = \beta \mathbf{R}_{t-1}^{(n)} + \bar{\mathbf{X}}_t^{(k)} \mathbf{W}_t^{(n)\top}. \tag{7.18}$$

Therefore, (7.16) can be rewritten as

$$\begin{aligned}
\mathbf{G}^{(n)} \mathbf{S}_t^{(n)} &= \beta \mathbf{R}_{t-1}^{(n)} + \mathbf{Y}_t^{(n)} \mathbf{W}_t^{(n)\top} \\
&= \beta \mathbf{G}_{t-1}^{(n)} \mathbf{S}_{t-1}^{(n)} + \mathbf{Y}_t^{(n)} \mathbf{W}_t^{(n)\top} \\
&= \mathbf{G}_{t-1}^{(n)} \mathbf{S}_t^{(n)} + \left(\mathbf{Y}_t^{(n)} - \mathbf{G}_{t-1}^{(n)} \mathbf{W}_t^{(n)}\right) \mathbf{W}_t^{(n)\top}. \tag{7.19}
\end{aligned}$$

Let the residual matrix $\Delta_t^{(n)}$ and coefficient matrix $\mathbf{V}_t^{(n)}$ be

$$\Delta_t^{(n)} = \mathbf{Y}_t^{(n)} - \mathbf{G}_{t-1}^{(n)}\mathbf{W}_t^{(n)}, \tag{7.20}$$

$$\mathbf{V}_t^{(n)} = \mathbf{W}_t^{(n)\top}\left(\mathbf{S}_t^{(n)}\right)^{-1}. \tag{7.21}$$

We obtain a simple rule for updating $\mathbf{G}_t^{(n)}$ as follows

$$\mathbf{G}_t^{(n)} = \mathbf{G}_{t-1}^{(n)} + \Delta_t^{(n)}\mathbf{V}_t^{(n)}. \tag{7.22}$$

After that, the TT-core $\boldsymbol{\mathcal{G}}_t^{(n)}$ will be derived from reshaping $\mathbf{G}_t^{(n)}$ into a 3-way tensor of size $r_{n-1} \times I_n \times r_n$.

We also note that when dealing with large-scale and high-rank tensors (i.e. $r_n \approx I_n$), TT-FOA can be sped up by using its stochastic approximation. We refer to this method as the stochastic TT-FOA. Particularly, the gradient $\nabla f(\mathbf{G}^{(n)})$ can be approximated by the instantaneous gradient of the last summand of $f(\mathbf{G}^{(n)})$. Thus, $\mathbf{S}_t^{(n)}$ can be computed by

$$\mathbf{S}_t^{(n)} \simeq \mathbf{W}_t^{(n)}(\mathbf{W}_t^{(n)})^\top. \tag{7.23}$$

Accordingly, the matrix $\mathbf{V}_t^{(n)}$ in (7.21) can be derived directly from the right inverse of $\mathbf{W}_t^{(n)}$. As a result, the stochastic TT-FOA not only skips several operations, but also saves a memory storage of $O(r_{n-1}^2 r_n^2)$ for storing $\mathbf{S}_t^{(n)}$ at time t. However, the stochastic approximation achieves a lower convergence rate than the original TT-FOA, see Fig. 7.7 for an illustration.

7.2.2.3   Computational Complexity and Memory Storage Analysis

For convenience of the analysis, we assume that the fixed dimensions of the tensor are equal to I while its TT-rank is $\mathbf{r}_{\mathrm{TT}} = [r, r, \ldots, r]$. In terms of computational complexity, TT-FOA first requires $O(|\Omega|r^2)$ flops for computing $\mathbf{g}_t^{(N)}$ by using the randomized LS method at time t. The cost for updating the k-th TT-core, $\boldsymbol{\mathcal{G}}_t^{(n)}$, comes from matrix-matrix products except an inverse operation for $\mathbf{S}_t^{(n)}$, hence it costs $O(I^{N-1}r^2)$ flops in general. It is due to that the matrix $\mathbf{S}_t^{(n)}$ is of size $r^2 \times r^2$, thus the computation of $(\mathbf{S}_t^{(n)})^{-1}$ is not expensive and independent of the tensor dimension. Therefore, the overall computational complexity is $O(I^{N-1}r^2)$. In term of memory storage, TT-FOA does not require to save the observation data at each time, it totally costs $O((N-1)(Ir^2 + r^4))$ words of memory for storing $n-1$ TT-cores and $N-1$ matrices $\mathbf{S}_t^{(n)}$. When the stochastic TT-FOA is applied, the memory storage is only $O((N-1)Ir^2)$ words of memory.

<div style="border-left: 6px solid #4472a8; padding-left: 10px;">

7.3   **Streaming Tensor-Train Decomposition with Missing Data**

</div>

In this subsection, we propose a novel adaptive algorithm called ATT (which stands for <u>A</u>daptive <u>T</u>ensor-<u>T</u>rain) 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 technically adept at estimating low-rank components of streaming tensors from noisy, imperfect, and incomplete observations as well as tracking their time variation in nonstationary environments. Besides, ATT can support parallel and distributed computing. To the best of our knowledge, ATT is the first TT algorithm which is capable of dealing with time-dependent streaming tensors with missing values.

7.3.1   Problem Formulation

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

$$\boldsymbol{Y}_t = \boldsymbol{P}_t \circledast \left(\boldsymbol{L}_t + \boldsymbol{N}_t \right). \tag{7.24}$$

Here, $\boldsymbol{P}_t$ is a binary (mask) tensor, $\boldsymbol{N}_t$ is a Gaussian noise tensor, and both tensors are of the same size with $\boldsymbol{X}_t$. The low-rank component $\boldsymbol{L}_t$ of $\boldsymbol{X}_t$ has the form

$$\boldsymbol{L}_t = \boldsymbol{G}_t^{(1)} \times_2^1 \boldsymbol{G}_t^{(2)} \times_3^1 \cdots \times_N^1 \mathbf{G}_t^{(N)}, \tag{7.25}$$

where $\boldsymbol{G}^{(n)} \in \mathbb{R}^{r_{n-1} \times I_n \times r_n}$ 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 $\boldsymbol{G}_t^{(N)}$, i.e., $\boldsymbol{G}_t^{(N)} = \left[\boldsymbol{G}_{t-1}^{(N)} \,\middle|\, \mathbf{G}_t^{(N)} \right]$.

Conventionally, TT-cores $\{\boldsymbol{G}_t^{(N)}\}_{n=1}^N$ can be obtained from:

$$\{\boldsymbol{G}_t^{(n)}\}_{n=1}^N = \operatorname*{argmin}_{\{\boldsymbol{G}^{(n)}\}_{n=1}^N} \left\| \hat{\boldsymbol{P}}_t \circledast \left(\boldsymbol{X}_t - \boldsymbol{G}^{(1)} \times_2^1 \boldsymbol{G}^{(2)} \times_3^1 \cdots \times_N^1 \boldsymbol{G}^{(N)} \right) \right\|_F^2, \tag{7.26}$$

where $\hat{\boldsymbol{\mathcal{P}}}_t$ is the observation mask of the underlying tensor $\boldsymbol{\mathcal{X}}_t$. In online settings, retaking the batch TT methods to solve (7.26) 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 $\boldsymbol{\mathcal{X}}_t$ in time.

Specifically, we propose to minimize the following exponentially weighted least-squares objective function, instead of (7.26):

$$\{\boldsymbol{\mathcal{G}}_t^{(n)}\}_{n=1}^N = \operatorname*{argmin}_{\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^N} \sum_{\tau=1}^t \beta^{t-\tau} \left\| \boldsymbol{\mathcal{P}}_\tau \circledast \left(\boldsymbol{\mathcal{Y}}_\tau - \boldsymbol{\mathcal{G}}^{(1)} \times_2^1 \cdots \times_{N-1}^1 \boldsymbol{\mathcal{G}}^{(N-1)} \right. \right.$$

$$\left. \left. \times_N^1 \mathrm{G}_\tau^{(N)} \right) \right\|_F^2 + \rho \sum_{n=1}^{N-1} \left\| \boldsymbol{\mathcal{G}}^{(n)} - \boldsymbol{\mathcal{G}}_{t-1}^{(n)} \right\|_F^2, \tag{7.27}$$

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. Note that, when $\beta = 1$ and $\rho = 0$, the objective function of (7.46) boils down to the batch one of (7.26).

To support our deployment in Section III, we make two mild assumptions on the data model: TT-cores $\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^{N-1}$ may either be static or vary slowly with time, i.e., $\boldsymbol{\mathcal{G}}_t^{(n)} \simeq \boldsymbol{\mathcal{G}}_{t-1}^{(n)}$; and TT-rank is supposed to be known.

7.3.2   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 (7.46) 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 10.

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

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

MAIN PROGRAM:

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

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

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

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

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

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

   $\bar{\mathbf{P}}_{t,i} = \text{diag}\{\boldsymbol{\mathcal{P}}_t(:, \ldots, :, i)\}$

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

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

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

  end

  $\boldsymbol{\mathcal{G}}_t^{(N)} = \begin{bmatrix} \boldsymbol{\mathcal{G}}_{t-1}^{(N)} & \mathbf{G}_t^{(N)} \end{bmatrix}$

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

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

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

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

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

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

   $\mathbf{B}_t^{(n)} = \text{reshape}\{\boldsymbol{\mathcal{B}}_t^{(n)}, [r_n, I_{n+1} I_{n+2} \ldots I_{N-1}]\}$

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

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

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

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

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

  end

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

end

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

Algorithm 10: ATT - Adaptive Tensor-Train Decomposition

7.3.2.1   Estimation of the temporal TT-core

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

$$\mathbf{G}_t^{(N)} = \underset{\mathbf{G}^{(N)}}{\arg\min} \left\| \boldsymbol{\mathcal{P}}_t \circledast \left(\boldsymbol{\mathcal{Y}}_t - \boldsymbol{\mathcal{H}}_{t-1} \times_N^1 \mathbf{G}^{(N)}\right) \right\|_F^2 + \lambda \left\| \mathbf{G}^{(N)} \right\|_F^2, \qquad (7.28)$$

where $\mathcal{H}_{t-1} = \mathcal{G}_{t-1}^{(1)} \times_2^1 \cdots \times_{N-1}^1 \mathcal{G}_{t-1}^{(N-1)}$ and $\lambda > 0$ is a small regularized parameter. Here, the first term of (7.28) is aimed at minimizing the residual error between observation and estimation for t-th temporal slice, while the introduction of $\lambda \|G^{(N)}\|_F^2$ is for avoiding the ill-posed computation in practice. Particularly, we can rewrite (7.28) as follows

$$G_t^{(N)} = \operatorname*{argmin}_{G^{(N)}} \left\| P_t \circledast \left(Y_t - H_{t-1} G^{(N)} \right) \right\|_2^2 + \lambda \left\| G^{(N)} \right\|_F^2, \qquad (7.29)$$

where $Y_t, P_t \in \mathbb{R}^{I_1 \dots I_{N-1} \times W}$, and $H_{t-1} \in \mathbb{R}^{I_1 \dots I_{N-1} \times r_{N-1}}$ are the unfolding matrices of $\mathcal{Y}_t$, $\mathcal{P}_t$ and $\mathcal{H}_{t-1}$, respectively. Furthermore, (7.29) can be decomposed into W subproblems w.r.t. W columns of $G^{(N)}$:

$$G_t^{(N)}(:, i) = \operatorname*{argmin}_{g_i} \left\| \bar{P}_{t,i} \left(y_{t,i} - H_{t-1} g_i \right) \right\|_2^2 + \lambda \|g_i\|_2^2. \qquad (7.30)$$

where $y_{t,i} = Y_t(:, i)$ and $\bar{P}_{t,i} = \mathtt{diag}\{P_t(:, i)\}$. The closed-form solution of the regularized least-squares (7.48) can be given by

$$G_t^{(N)}(:, i) = \left(H_{t-1}^\top \bar{P}_{t,i} H_{t-1} + \lambda I_{r_{N-1}} \right)^{-1} H_{t-1}^\top \bar{P}_{t,i} y_{t,i}. \qquad (7.31)$$

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

7.3.2.2   Estimation of the non-temporal TT-cores

We update $\{\mathcal{G}^{(n)}\}_{n=1}^{N-1}$ by minimizing

$$\mathcal{G}_t^{(n)} = \operatorname*{argmin}_{\mathcal{G}^{(n)}} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \mathcal{P}_\tau \circledast \left(\mathcal{Y}_\tau - \mathcal{A}_{t-1}^{(n)} \times_n^1 \mathcal{G}^{(n)} \times_{n+1}^1 \mathcal{B}_\tau^{(n)} \right) \right\|_F^2$$
$$+ \rho \left\| \mathcal{G}^{(n)} - \mathcal{G}_{t-1}^{(n)} \right\|_F^2, \qquad (7.32)$$

where $\mathcal{A}_{t-1}^{(n)} = \mathcal{G}_{t-1}^{(1)} \times_2^1 \cdots \times_{n-1}^1 \mathcal{G}_{t-1}^{(n-1)}$ and $\mathcal{B}_\tau^{(n)} = \mathcal{G}_{t-1}^{(n+1)} \times_{n+2}^1 \cdots \times_{N-1}^1 \mathcal{G}_{t-1}^{(N-1)} \times_N^1 G_\tau^{(N)}$. For a better interpretation, we further recast (7.32) as

$$G_t^{(n)} = \operatorname*{argmin}_{G^{(n)}} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \underline{P}_\tau^{(n)} \circledast \left(\underline{Y}_\tau^{(n)} - G^{(n)} W_\tau^{(n)} \right) \right\|_F^2 + \rho \left\| G^{(n)} - G_{t-1}^{(n)} \right\|_F^2,$$
$$(7.33)$$

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

$$\mathbf{A}_{t-1}^{(n)} = \texttt{reshape}\{\boldsymbol{\mathcal{A}}_{t-1}^{(n)}, [r_{n-1}, I_1 I_2 \dots I_{n-1}]\} \tag{7.34}$$

$$\mathbf{B}_\tau^{(n)} = \texttt{reshape}\{\boldsymbol{\mathcal{B}}_t^{(n)}, [r_n, I_{n+1} I_{n+2} \dots I_{N-1}]\} \tag{7.35}$$

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 $\mathbf{G}_t^{(n)}$ as follows:

$$\mathbf{g}_{t,m}^{(n)} = \underset{\mathbf{g}_m^{(n)}}{\text{argmin}} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \bar{\mathbf{P}}_{\tau,m}^{(n)} \left(\mathbf{y}_{\tau,m}^{(n)} - \mathbf{g}_m^{(n)} \mathbf{W}_\tau^{(n)} \right)^\top \right\|_2^2 + \rho \left\| \mathbf{g}_m^{(n)} - \mathbf{g}_{t-1,m}^{(n)} \right\|_2^2, \tag{7.36}$$

where $\mathbf{y}_{\tau,m}^{(n)} = \underline{\mathbf{Y}}_\tau^{(n)}(m,:)$ and $\bar{\mathbf{P}}_{\tau,m}^{(n)} = \texttt{diag}\{\underline{\mathbf{P}}_\tau^{(n)}(m,:)\}$.

Specifically, $\mathbf{g}_{t,m}^{(n)}$ can be derived from setting the gradient of the function in (7.36) to zero:

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

$$= \rho \left(\mathbf{g}_{t-1,m}^{(n)} \right)^\top + \sum_{\tau=1}^{t} \beta^{t-\tau} \mathbf{W}_\tau^{(n)} \bar{\mathbf{P}}_{\tau,m}^{(n)} \left(\mathbf{y}_{\tau,m}^{(n)} \right)^\top. \tag{7.37}$$

The closed-form solution of (7.37) is then given by

$$\mathbf{g}_{t,m}^{(n)} = \left[\left(\mathbf{S}_{t,m}^{(n)} + \rho \mathbf{I}_{r_{n-1}r_n} \right)^{-1} \left(\mathbf{d}_{t,m}^{(n)} + \rho \left(\mathbf{g}_{t-1,m}^{(n)} \right)^\top \right) \right]^\top, \tag{7.38}$$

where $\mathbf{S}_{t,m}^{(n)}$ and $\mathbf{d}_{t,m}^{(n)}$ can be recursively updated as

$$\mathbf{S}_{t,m}^{(n)} = \beta \mathbf{S}_{t-1,m}^{(n)} + \mathbf{W}_t^{(n)} \bar{\mathbf{P}}_{t,m}^{(n)} \left(\mathbf{W}_t^{(n)} \right)^\top \tag{7.39}$$

$$\mathbf{d}_{t,m}^{(n)} = \beta \mathbf{d}_{t-1,m}^{(n)} + \mathbf{W}_t^{(n)} \bar{\mathbf{P}}_{t,m}^{(n)} \left(\mathbf{y}_{t,m}^{(n)} \right)^\top. \tag{7.40}$$

After doing some simple calculations, we can rewrite (7.38) as

$$\mathbf{g}_{t,m}^{(n)} = \mathbf{g}_{t-1,m}^{(n)} + \left(\boldsymbol{\delta} \mathbf{y}_{t,m}^{(n)} \bar{\mathbf{P}}_{t,m}^{(n)} \left(\mathbf{W}_t^{(n)} \right)^\top + \beta \rho \boldsymbol{\delta} \mathbf{g}_{t-1,m}^{(n)} \right) \left(\mathbf{S}_{t,m}^{(n)} + \rho \mathbf{I}_{r_{n-1}r_n} \right)^{-\top}, \tag{7.41}$$

where $\boldsymbol{\delta} \mathbf{y}_{t,m}^{(n)} = \bar{\mathbf{P}}_{t,m}^{(n)} \left(\mathbf{y}_{t,m}^{(n)} - \mathbf{g}_{t-1,m}^{(n)} \mathbf{W}_t^{(n)} \right)^\top$ and $\boldsymbol{\delta} \mathbf{g}_{t-1,m}^{(n)} = \mathbf{g}_{t-1,m}^{(n)} - \mathbf{g}_{t-2,m}^{(n)}$. Accordingly, a recursive rule with a lower space complexity for updating the whole matrix $\mathbf{G}_t^{(n)}$ at the same time can be given by

$$\mathbf{G}_t^{(n)} = \mathbf{G}_{t-1}^{(n)} + \left(\left(\underline{\mathbf{P}}_t^{(n)} \circledast \Delta \underline{\mathbf{Y}}_t^{(n)} \right) \left(\mathbf{W}_t^{(n)} \right)^\top + \rho \Delta \mathbf{G}_{t-1}^{(n)} \right) \left(\mathbf{S}_t^{(n)} + \rho \mathbf{I}_{r_{n-1}r_n} \right)^{-\top}, \tag{7.42}$$

where $\Delta \underline{\mathbf{Y}}_{t,m}^{(n)} = \underline{\mathbf{Y}}_t^{(n)} - \mathbf{G}_{t-1}^{(n)}\mathbf{W}_t^{(n)}$ and $\Delta \mathbf{G}_{t-1}^{(n)} = \mathbf{G}_{t-1}^{(n)} - \mathbf{G}_{t-2}^{(n)}$.

Then, we simply set $\boldsymbol{\mathcal{G}}_t^{(n)} = \texttt{reshape}\{\mathbf{G}_t^{(n)}, [r_{n-1}, I_n, r_n]\}$. The rule (7.42) also suggests that we can incrementally update $\{\boldsymbol{\mathcal{G}}_t^{(n)}\}_{n=1}^{N-1}$ in parallel without disrupting other each. In other words, ATT can support parallel and distributed computing.

7.3.2.3 Complexity 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 $\boldsymbol{O}(W|\Omega_t|r^2)$ 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 $\boldsymbol{O}\big((N-1)I^{N-1}r^4\big)$ flops. The overall complexity of ATT is $\boldsymbol{O}\big(r^2 \max\big\{(N-1)I^{N-1}r^2, W|\Omega_t|\big\}\big)$ flops. In term of memory storage, ATT needs $\boldsymbol{O}\big((N-1)(2Ir^2 + r^4)\big)$ words of memory for storing $\big\{\boldsymbol{\mathcal{G}}_t^{(n)}\big\}_{n=1}^{N-1}$, $\big\{\Delta\boldsymbol{\mathcal{G}}_t^{(n)}\big\}_{n=1}^{N-1}$, and $\big\{\mathbf{S}_t^{(n)}\big\}_{n=1}^{N-1}$.

Compared to batch TT methods (e.g., TT-SVD [16] and TT-HSVD [363]), 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.

7.4 Streaming Tensor-Train Decomposition with Sparse Outliers

In this paper, we introduce a new tensor-train method for factorizing incomplete high-order streaming tensors possibly corrupted by sparse outliers. The proposed method is referred to as ROBOT which stands for <u>ROB</u>ust <u>O</u>nline <u>T</u>ensor-Train decomposition. ROBOT involves two well-known optimization methods: block-coordinate descent (BCD) and recursive least-squares (RLS). Thanks to the BCD framework, ROBOT decomposes the main optimization into two stages: (i) online outlier rejection and (ii) tracking of TT-cores in time. In the former stage, we apply an effective ADMM solver to estimate the last (temporal) TT-core and sparse outliers living in observations. In the latter stage, we present an efficient RLS solver to minimize an exponential weighted least-squares objective function accounting for missing entries and time variations of TT-cores. Technically, ROBOT is capable of estimating the
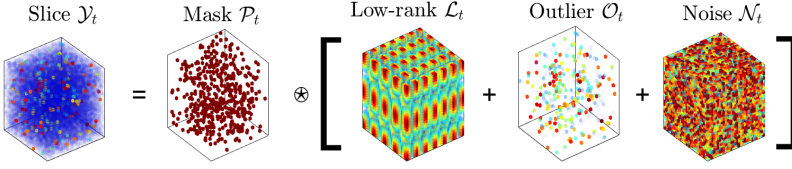
Figure 7.3: Temporal slice $\mathcal{Y}_t$ with missing data and outliers.

low-rank components of the underlying tensor from imperfect streams (i.e., due to noise, outliers, and missing data) and tracking their time variation in dynamic environments. To the best of our knowledge, ROBOT is the first streaming TT decomposition robust to sparse outliers, missing data, and time variation.

7.4.1 Problem Formulation

In this paper, we study the robust adaptive tensor-train decomposition of a N-order streaming tensor $\mathcal{X}_t$ in the presence of both sparse outliers and missing data. Without loss of generality, we suppose the last dimension of $\mathcal{X}_t$ is temporal, while the others remain constant with time, i.e., $\mathcal{X}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times I_N^t}$. Specifically, at time t, $\mathcal{X}_t$ is obtained by concatenating the incoming data stream $\mathcal{Y}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times W}$ (with $W \geq 1$) to the old observation $\mathcal{X}_{t-1}$ along the temporal dimension I_N^t, i.e.,

$$\mathcal{X}_t = \mathcal{X}_{t-1} \boxplus_N \mathcal{Y}_t \quad \text{and} \quad I_N^t = I_N^{t-1} + W. \tag{7.43}$$

The temporal slice $\mathcal{Y}_t$ is supposed to have the form

$$\mathcal{Y}_t = \mathcal{P}_t \circledast \left(\mathcal{L}_t + \mathcal{O}_t + \mathcal{N}_t \right), \tag{7.44}$$

see Fig. 7.3 for an illustration. Particularly, $\mathcal{P}_t$ is a binary mask tensor, $\mathcal{O}_t$ is a sparse outlier tensor, $\mathcal{N}_t$ is a Gaussian noise tensor, and they share the same size as $\mathcal{Y}_t$. The low-rank component $\mathcal{L}_t$ of $\mathcal{Y}_t$ is expressed as

$$\mathcal{L}_t = \mathcal{G}_t^{(1)} \times_2^1 \mathcal{G}_t^{(2)} \times_3^1 \cdots \times_N^1 \mathbf{G}_t^{(N)}, \tag{7.45}$$

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; $[r_1, r_2, \ldots, r_{N-1}]$ is called TT-rank; and $\mathbf{G}_t^{(N)} \in \mathbb{R}^{r_{N-1} \times W}$ contains the last W columns of $\mathcal{G}_t^{(N)}$.

In online settings, we propose to minimize the following objective func-

tion:

$$\underset{\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^{N},\boldsymbol{O}}{\operatorname{argmin}} \sum_{k=1}^{t} \beta^{t-k} \left(\left\| \boldsymbol{\mathcal{P}}_k \circledast \left(\boldsymbol{\mathcal{G}}^{(1)} \times_2^1 \cdots \times_{N-1}^1 \boldsymbol{\mathcal{G}}^{(N-1)} \times_N^1 \mathbf{G}_\tau^{(n)} + \boldsymbol{O}_k - \boldsymbol{\mathcal{Y}}_k \right) \right\|_F^2 \right.$$

$$\left. + \rho_1 \|\boldsymbol{O}_k\|_1 \right) + \rho_2 \sum_{n=1}^{N-1} \left\| \boldsymbol{\mathcal{G}}^{(n)} - \boldsymbol{\mathcal{G}}_{t-1}^{(n)} \right\|_F^2. \tag{7.46}$$

Here, $\beta \in (0, 1]$ plays the role of a forgetting factor in adaptive filter theory which aims to reduce the impact of distant observations as well as deal with nonstationary environments [364]. The ℓ_1-norm enforces the sparsity on $\boldsymbol{O}$ (the outliers), while the last regularization term of (7.46) is to control the time variation of TT-cores between two consecutive instances. In addition, we make two mild assumptions on the data model to support our algorithm development in Section III: TT-cores $\{\boldsymbol{\mathcal{G}}^{(n)}\}_{n=1}^{N-1}$ may either be static or vary slowly with time, i.e., $\boldsymbol{\mathcal{G}}_t^{(n)} \simeq \boldsymbol{\mathcal{G}}_{t-1}^{(n)}$; and the TT-rank is supposed to be known.

7.4.2 Proposed Method

In this section, we propose an adaptive method called ROBOT (which stands for <u>ROB</u>ust <u>O</u>nline <u>T</u>ensor-Train) for factorizing tensors derived from data streams in the presence of sparse outliers and missing data. Particularly, we decompose the main problem (7.46) into two stages:

- Stage 1: update $\boldsymbol{\mathcal{G}}_t^{(N)}$ and $\boldsymbol{O}_t$ given $\{\boldsymbol{\mathcal{G}}_{t-1}^{(n)}\}_{n=1}^{N-1}$;

- Stage 2: estimate $\boldsymbol{\mathcal{G}}_t^{(n)}$ given $\boldsymbol{\mathcal{G}}_t^{(N)}$, $\boldsymbol{O}_t$, and the remaining TT-cores, for $n = 1, 2, \ldots, N-1$.

7.4.2.1 Estimation of the temporal TT-core and Outlier

At each time t, we estimate $\mathbf{G}_t^{(N)}$ and $\boldsymbol{O}_t$ by solving

$$\{\mathbf{G}_t^{(N)}, \boldsymbol{O}_t\} = \underset{\mathbf{G}^{(N)}, \boldsymbol{O}}{\operatorname{argmin}} \left\| \boldsymbol{\mathcal{P}}_t \circledast \left(\boldsymbol{\mathcal{H}}_{t-1} \times_N^1 \mathbf{G}^{(N)} + \boldsymbol{O} - \boldsymbol{\mathcal{Y}}_t \right) \right\|_F^2 + \rho_1 \|\boldsymbol{O}\|_1$$

$$+ \rho_2 \|\mathbf{G}^{(N)}\|_F^2, \tag{7.47}$$

where $\boldsymbol{\mathcal{H}}_{t-1} = \boldsymbol{\mathcal{G}}_{t-1}^{(1)} \times_2^1 \cdots \times_{N-1}^1 \boldsymbol{\mathcal{G}}_{t-1}^{(N-1)}$ and the term $\rho_2 \|\mathbf{G}^{(N)}\|_F^2$ is to mitigate ill matrix conditions. Interestingly, we exploit the fact that (7.47) can be decomposed into W sub-problems w.r.t. W columns of $\mathbf{G}_t^{(N)}$, as follows:

$$\underset{\mathbf{g}_i, \mathbf{o}_i}{\operatorname{argmin}} \left\| \underline{\mathbf{P}}_{t,i} \left(\mathbf{H}_{t-1}\mathbf{g}_i + \mathbf{o}_i - \mathbf{y}_{t,i} \right) \right\|_2^2 + \rho_1 \|\mathbf{o}_i\|_1 + \rho_2 \|\mathbf{g}_i\|_2^2. \tag{7.48}$$

Here, $\mathbf{g}_i$, $\mathbf{o}_i$, and $\mathbf{y}_{t,i}$ are, respectively, the i-th column of $\mathbf{G}^{(N)}$, the two unfolding matrices of $\boldsymbol{O}$ and $\boldsymbol{\mathcal{Y}}_t$; the mask $\underline{\mathbf{P}}_{t,i} = \mathtt{diag}\{\underline{\mathbf{P}}_t^{(N)}(i,:)\}$; while the matrix $\mathbf{H}_{t-1} \in \mathbb{R}^{I_1 \dots I_{N-1} \times r_{N-1}}$ is a matricization of $\mathcal{H}_{t-1}$.

Since both ℓ_1-norm and ℓ_2-norm are convex, (7.48) can be effectively minimized by several methods, e.g., block coordinate descent (BCD) [365] and alternating direction method of multipliers (ADMM) [114]. In this work, we adopt the ADMM solver introduced in our companion work on robust subspace tracking [25]. Specifically, the update rule at the j-th iteration of the solver is given by

$$\mathbf{g}^j = \left(\mathbf{H}_{t-1}^\top \underline{\mathbf{P}}_{t,i} \mathbf{H}_{t-1} + \rho_2 \mathbf{I}_{r_{N-1}}\right)^{-1} \mathbf{H}_{t-1}^\top \underline{\mathbf{P}}_{t,i} \left(\mathbf{y}_{t,i} - \mathbf{o}^{j-1} + \mathbf{e}^{j-1}\right),$$

$$\mathbf{z}^j = \underline{\mathbf{P}}_{t,i} \left(\mathbf{H}_{t-1}\mathbf{g}^j + \mathbf{s}^{j-1} - \mathbf{y}_{t,i}\right),$$

$$\mathbf{e}^j = \frac{\lambda_1}{1 + \lambda_1} \mathbf{z}^j + \frac{1}{1 + \lambda_1} \mathcal{S}_{1 + \frac{1}{\lambda_1}}(\mathbf{z}^j),$$

$$\mathbf{u}^j = \frac{1}{1 + \lambda_2}\left(\mathbf{P}_{t,i}\left(\mathbf{y}_{t,i} - \mathbf{H}_{t-1}\mathbf{g}^j\right)\right) - \lambda_2(\mathbf{o}^{j-1} - \mathbf{r}^{j-1}),$$

$$\mathbf{o}^j = \mathcal{S}_{\rho_1/\lambda_2}(\mathbf{u}^j + \mathbf{r}^{j-1}),$$

$$\mathbf{r}^j = \mathbf{r}^{j-1} + \mathbf{u}^j - \mathbf{s}^j.$$

Here, $\{\mathbf{z}^j, \mathbf{e}^j, \mathbf{u}^j, \mathbf{r}^j\}$ are dummy variables aiming to accelerate the update initialized as zeros; the augmented Lagrangian parameters λ_1 and λ_2 can be chosen in the range $[1, 1.8]$; and $\mathcal{S}_\alpha(.)$ is the soft-thresholding operator defined as $\mathcal{S}_\alpha(x) = \max(0, x - \alpha) - \max(0, -x - \alpha)$. We refer the readers to [25] for further details. Note that since (7.48) is a biconvex minimization problem, and thus, we can apply any other existing proved algorithm to obtain its optimal solution [366].

The temporal TT-core $\boldsymbol{\mathcal{G}}_t^{(N)}$ is simply obtained by $\boldsymbol{\mathcal{G}}_t^{(N)} = [\boldsymbol{\mathcal{G}}_{t-1}^{(N)} \; \mathbf{G}_t^{(N)}]$. In addition, we can re-update $\mathbf{G}_t^{(N)}$ in the same way as above when others TT-cores $\{\boldsymbol{\mathcal{G}}_t^{(n)}\}_{n=1}^{N-1}$ are updated. Furthermore, after obtaining the outlier $\boldsymbol{O}_t$, we can accelerate the tracking ability of ROBOT by re-updating the observation mask $\boldsymbol{\mathcal{P}}_t$ as follows

$$\left[\widetilde{\boldsymbol{\mathcal{P}}}_t\right]_{i_1 i_2 \dots i_N} = \begin{cases} 0, & \text{if } \left[\boldsymbol{O}_t\right]_{i_1 i_2 \dots i_N} \neq 0, \\ \left[\boldsymbol{\mathcal{P}}_t\right]_{i_1 i_2 \dots i_N}, & \text{otherwise.} \end{cases} \tag{7.49}$$

It is motivated by the following observation: In the literature of robust subspace tracking (RST), the outlier rejection step can facilitate the tracking ability of RST estimators because only "clean" data are involved in the tracking process [25]. Our stage 2 for tracking the TT-cores can be viewed as an extended version of RST for high-order streaming tensors, so the outlier rejection mechanism of (7.49) can improve its performance.

7.4.2.2   Estimation of TT-cores

We estimate $\{\mathcal{G}^{(n)}\}_{n=1}^{N-1}$ by minimizing

$$\mathcal{G}_t^{(n)} = \underset{\mathcal{G}^{(n)}}{\arg\min} \sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \widetilde{\mathcal{P}}_\tau \circledast \left(\mathcal{A}_{t-1}^{(n)} \times_n^1 \mathcal{G}^{(n)} \times_{n+1}^1 \mathcal{B}_\tau^{(n)} - \mathcal{Y}_\tau \right) \right\|_F^2$$

$$+ \rho_2 \left\| \mathcal{G}^{(n)} - \mathcal{G}_{t-1}^{(n)} \right\|_F^2, \tag{7.50}$$

where $\mathcal{A}_{t-1}^{(n)} = \mathcal{G}_{t-1}^{(1)} \times_2^1 \cdots \times_{n-1}^1 \mathcal{G}_{t-1}^{(n-1)}$ and $\mathcal{B}_\tau^{(n)} = \mathcal{G}_{t-1}^{(n+1)} \times_{n+2}^1 \cdots \times_{N-1}^1 \mathcal{G}_{t-1}^{(N-1)} \times_N^1 \mathcal{G}_\tau^{(n)}$ while the term $\mathcal{O}_k$ is discarded due to outlier rejection mechanism (7.49), i.e., $\widetilde{\mathcal{P}}_t \circledast (\mathcal{Y}_t - \mathcal{O}_t) = \widetilde{\mathcal{P}}_t \circledast \mathcal{Y}_t$. Particularly, (7.50) can be regarded as the optimization problem of adaptive TT decomposition from incomplete observations $\{\mathcal{Y}_k\}_{k=1}^t$ with new binary masks $\{\widetilde{\mathcal{P}}_k\}_{k=1}^t$. Accordingly, we can apply the effective recursive least-squares (RLS) method as proposed in our work [31] for minimizing (7.50). For the sake of completeness, we describe here the main steps of the RLS solver and refer the readers to [31] for further details.

For a better interpretation, we first recast (7.50) as

$$\mathbf{G}_t^{(n)} = \underset{\mathbf{G}^{(n)}}{\arg\min} \sum_{m=1}^{I_n} \left(\sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \bar{\mathbf{P}}_{\tau,m}^{(n)} \left(\mathbf{g}_m^{(n)} \left(\mathbf{B}_\tau^{(n)} \otimes \mathbf{A}_{t-1}^{(n)} \right) - \mathbf{y}_{\tau,m}^{(n)} \right) \right\|_2^2 \right.$$

$$\left. + \rho_2 \left\| \mathbf{g}_m^{(n)} - \mathbf{g}_{t-1,m}^{(n)} \right\|_2^2 \right), \tag{7.51}$$

where $\mathbf{g}_m^{(n)}$ is the m-th row of $\mathbf{G}^{(n)} \in \mathbb{R}^{I_n \times r_{n-1} r_n}$ which is the transpose of the mode-2 unfolding matrix of $\mathcal{G}^{(n)}$, $\bar{\mathbf{P}}_{\tau,m} = \mathtt{diag}\{\underline{\tilde{\mathbf{P}}}_\tau^{(n)}(m,:)\}$, and

$$\mathbf{A}_{t-1}^{(n)} = \mathtt{reshape}\{\mathcal{A}_{t-1}^{(n)}, [r_{n-1}, I_1 I_2 \ldots I_{n-1}]\}, \tag{7.52}$$

$$\mathbf{B}_\tau^{(n)} = \mathtt{reshape}\{\mathcal{B}_t^{(n)}, [r_n, I_{n+1} I_{n+2} \ldots I_{N-1}]\}. \tag{7.53}$$

Let us denote $\mathbf{W}_\tau^{(n)} = \mathbf{B}_\tau^{(n)} \otimes \mathbf{A}_{t-1}^{(n)}$ and

$$\mathbf{S}_{\tau,m}^{(n)} = \sum_{\tau=1}^{t} \beta^{t-\tau} \mathbf{W}_t^{(n)} \bar{\mathbf{P}}_{t,m}^{(n)} \left(\mathbf{W}_t^{(n)} \right)^\top, \tag{7.54}$$

$$\mathbf{d}_{t,m}^{(n)} = \sum_{\tau=1}^{t} \beta^{t-\tau} \mathbf{W}_\tau^{(n)} \bar{\mathbf{P}}_{\tau,m}^{(n)} \left(\mathbf{y}_{\tau,m}^{(n)} \right)^\top. \tag{7.55}$$

At time t, we then have

$$\mathbf{S}_{t,m}^{(n)} = \beta \mathbf{S}_{t-1,m}^{(n)} + \mathbf{W}_t^{(n)} \bar{\mathbf{P}}_{t,m}^{(n)} \left(\mathbf{W}_t^{(n)} \right)^\top \tag{7.56}$$

$$\mathbf{d}_{t,m}^{(n)} = \beta \mathbf{d}_{t-1,m}^{(n)} + \mathbf{W}_t^{(n)} \bar{\mathbf{P}}_{t,m}^{(n)} \left(\mathbf{y}_{t,m}^{(n)} \right)^\top. \tag{7.57}$$

Setting the gradient of (7.51) to zero results in:

$$\sum_{m=1}^{I_n} \left(\mathbf{S}_{t,m}^{(n)} + \rho_2 \mathbf{I}_{r_{n-1} r_n}\right) \left(\mathbf{g}_m^{(n)}\right)^{\top} = \sum_{m=1}^{I_n} \left(\mathbf{d}_{t,m}^{(n)} + \rho_2 \left(\mathbf{g}_{t-1,m}^{(n)}\right)^{\top}\right). \tag{7.58}$$

Therefore, we can express each row $\mathbf{g}_{t,m}^{(n)}$ of $\mathbf{G}_t^{(n)}$ separately as

$$\left(\mathbf{S}_{t,m}^{(n)} + \rho_2 \mathbf{I}_{r_{n-1} r_n}\right) \left(\mathbf{g}_{t,m}^{(n)}\right)^{\top} = \mathbf{d}_{t,m}^{(n)} + \rho_2 \left(\mathbf{g}_{t-1,m}^{(n)}\right)^{\top}. \tag{7.59}$$

Thanks to (7.56) and (7.57), we further recast (7.59) as

$$\mathbf{g}_{t,m}^{(n)} = \mathbf{g}_{t-1,m}^{(n)} + \left(\boldsymbol{\delta}\mathbf{y}_{t,m}^{(n)} \bar{\mathbf{P}}_{t,m}^{(n)} \left(\mathbf{W}_t^{(n)}\right)^{\top} + \beta \rho_2 \boldsymbol{\delta}\mathbf{g}_{t-1,m}^{(n)}\right) \left(\mathbf{S}_{t,m}^{(n)} + \rho_2 \mathbf{I}_{r_{n-1} r_n}\right)^{-\top}, \tag{7.60}$$

where $\boldsymbol{\delta}\mathbf{y}_{t,m}^{(n)} = \bar{\mathbf{P}}_{t,m}^{(n)} \left(\mathbf{y}_{t,m}^{(n)} - \mathbf{g}_{t-1,m}^{(n)} \mathbf{W}_t^{(n)}\right)^{\top}$ and $\boldsymbol{\delta}\mathbf{g}_{t-1,m}^{(n)} = \mathbf{g}_{t-1,m}^{(n)} - \mathbf{g}_{t-2,m}^{(n)}$. Collecting all rows $\mathbf{g}_{t,m}^{(n)}$ together (for $m = 1, 2, \ldots, I_n$), we obtain a simpler recursive rule as

$$\mathbf{G}_t^{(n)} = \mathbf{G}_{t-1}^{(n)} + \left(\left(\underline{\mathbf{P}}_t^{(n)} \circledast \Delta \underline{\mathbf{X}}_t^{(n)}\right) \left(\mathbf{W}_t^{(n)}\right)^{\top} + \beta \rho_2 \Delta \mathbf{G}_{t-1}^{(n)}\right) \left(\mathbf{S}_t^{(n)} + \rho_2 \mathbf{I}_{r_{n-1} r_n}\right)^{-\top}, \tag{7.61}$$

where $\Delta \mathbf{X}_{t,m}^{(n)} = \underline{\mathbf{X}}_t^{(n)} - \mathbf{G}_{t-1}^{(n)} \mathbf{W}_t^{(n)}$ and $\Delta \mathbf{G}_{t-1}^{(n)} = \mathbf{G}_{t-1}^{(n)} - \mathbf{G}_{t-2}^{(n)}$, and $\mathbf{S}_t^{(n)} = \beta \mathbf{S}_{t-1}^{(n)} + \mathbf{W}_t^{(n)} \left(\mathbf{W}_t^{(n)}\right)^{\top}$. To enable the recursive update (7.61), we set $\Delta \mathbf{G}_0^{(n)} = \mathbf{0}$ and $\mathbf{S}_0^{(n)} = \delta^{(n)} \mathbf{I}_{r_{n-1} r_n}$ with $\delta^{(n)} > 0$.

7.4.2.3 Computational Complexity and Memory Storage

For short, we suppose $I_n = I$ and $r_n = r$ for all $n = 1, 2, \ldots, N - 1$. In Stage 1, ROBOT requires a cost of $\boldsymbol{O}(W|\Omega_t|r^2)$ flops for estimating both $\mathbf{G}_t^{(N)}$ and $\boldsymbol{O}_t$ where $|\Omega_t|$ denotes the number of observed data in $\boldsymbol{\mathcal{Y}}_t$. In Stage 2, ROBOT needs a cost of $\boldsymbol{O}\left((N-1)I^{N-1}r^4\right)$ flops for tracking $N - 1$ TT-cores $\{\boldsymbol{\mathcal{G}}_t^{(n)}\}_{n=1}^{N-1}$. Therefore, the overall complexity of ROBOT is $\boldsymbol{O}\left(r^2 \max\left\{(N-1)I^{N-1}r^2, W|\Omega_t|\right\}\right)$ flops. With respect to memory storage, ROBOT requires $\boldsymbol{O}\left((N-1)(2Ir^2 + r^4)\right)$ words of memory for storing $\{\boldsymbol{\mathcal{G}}_t^{(n)}\}_{n=1}^{N-1}$, $\{\Delta \boldsymbol{\mathcal{G}}_t^{(n)}\}_{n=1}^{N-1}$, and $\{\mathbf{S}_t^{(n)}\}_{n=1}^{N-1}$.

7.5 Experiments

In this section, we conduct several experiments on both synthetic and real data to evaluate the performance of TT-FOA, ATT, and ROBOT for adaptive
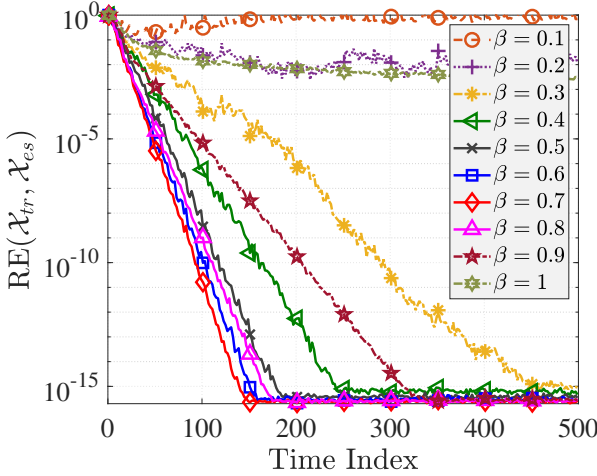
Figure 7.4: Effect of the forgetting factor β on the performance of TT-FOA.

TT decomposition. Experiments are implemented in MATLAB platform and are available online to facilitate replicability and reproducibility.[1]

7.5.1   Performance of TT-FOA

We investigate the tracking ability of TT-FOA with respect to the following aspects: effect of the forgetting factor λ, effect of the noise level σ, its performance in time-varying environments, and its use for real data.

7.5.1.1   Synthetic Data

We generate streaming 4-way tensors $\boldsymbol{X}_t \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4^t}$ of a TT-rank vector $\mathbf{r}_{\mathrm{TT}} = [r_1, r_2, r_3]$ as follows:

$$\boldsymbol{Y}_t = \boldsymbol{G}_t^{(1)} \times_2^1 \boldsymbol{G}_t^{(2)} \times_3^1 \boldsymbol{G}_t^{(3)} \times_4^1 \mathbf{g}_t^{(4)} + \epsilon \boldsymbol{N}_t,$$

where the 3-way tensor $\boldsymbol{Y}_t \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is the t-th slice of $\boldsymbol{X}_t$; $\boldsymbol{N}_t$ is a Gaussian noise tensor of the same size with $\boldsymbol{Y}_t$ and ϵ controls the noise level; the last column $\mathbf{g}_t^{(4)}$ of TT-core $\boldsymbol{G}_t^{(4)}$ is a random vector living on $\mathbb{R}^{r_3}$ space; TT-cores $\boldsymbol{G}_t^{(1)}, \boldsymbol{G}_t^{(2)}$ and $\boldsymbol{G}_t^{(3)}$ are, respectively, of size $I_1 \times r_1$, $r_1 \times I_2 \times r_2$ and $r_2 \times I_3 \times r_3$ given by

$$\boldsymbol{G}_t^{(n)} = (1 - \sigma)\boldsymbol{G}_{t-1}^{(n)} + \sigma \boldsymbol{N}_t^{(n)},$$

---

[1]https://github.com/thanhtbt/ATT & https://github.com/thanhtbt/ATT-miss & https://github.com/thanhtbt/ROBOT
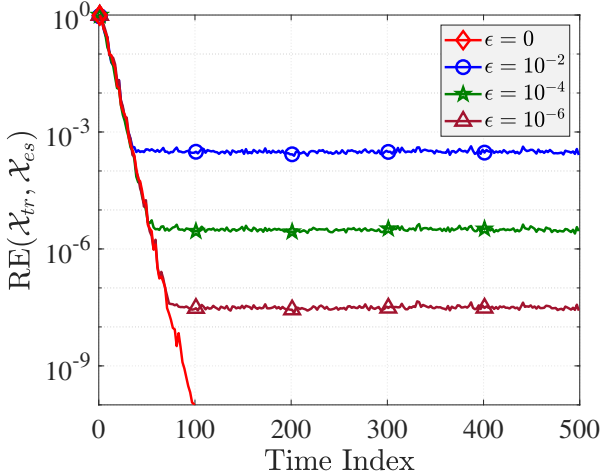
Figure 7.5: Effect of the noise level ϵ on the performance of TT-FOA.

where σ controls the variation of the TT-cores between two consecutive instances, $\boldsymbol{\mathcal{N}}_1^{(n)} \in \mathbb{R}^{I_1 \times r_1}$ and $\boldsymbol{\mathcal{N}}_t^{(n)} \in \mathbb{R}^{r_{n-1} \times I_n \times r_n}$ are noise tensors whose entries are i.i.d from the Gaussian distribution with zero-mean and unit-variance.

To measure the estimation accuracy, we use the relative error (RE) metric given by

$$\mathrm{RE}(\boldsymbol{\mathcal{X}}_{tr}, \boldsymbol{\mathcal{X}}_{es}) = \frac{\left\| \boldsymbol{\mathcal{X}}_{tr} - \boldsymbol{\mathcal{X}}_{es} \right\|_F}{\left\| \boldsymbol{\mathcal{X}}_{tr} \right\|_F}, \tag{7.62}$$

where $\boldsymbol{\mathcal{X}}_{tr}$ (resp. $\boldsymbol{\mathcal{X}}_{es}$) refers to the true tensor (resp. estimated tensor).

The choice of forgetting factor λ plays a central role in how fast TT-FOA converges. Fig. 7.4 shows the experimental results of applying the algorithm to a static and free-noise tensor whose size is $10 \times 12 \times 15 \times 500$ and its TT-rank is $\mathbf{r}_{\mathrm{TT}} = [2, 3, 5]$. We can see that the relative error is minimized when λ is round 0.7. TT-FOA fails when λ is close to its infimum or supremum. We then fix $\lambda = 0.7$ in the next experiments.

To study the effect of noise on the performance of our algorithm, we vary the value of the noise level ϵ and access its estimation on the same tensor above. The result is shown in Fig. 7.11. When we reduce the noise, relative error (RE) between the ground truth and estimation degrades gradually and converges towards a steady state error bound. Note that the convergence rate of the algorithm is not affected by the noise level but only its estimation error.

We next consider a scenario where TT-cores change slowly with time and abruptly at instant $t = 300$. Fig. 7.6 shows the performance of TT-FOA applying to the same free-noise tensor versus the time-varying factor σ. In
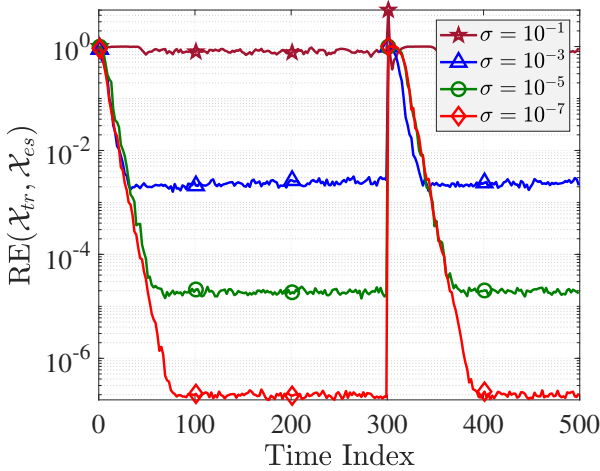
Figure 7.6: Effect of the time-varying factor σ on the performance of TT-FOA in the case of noise-free.
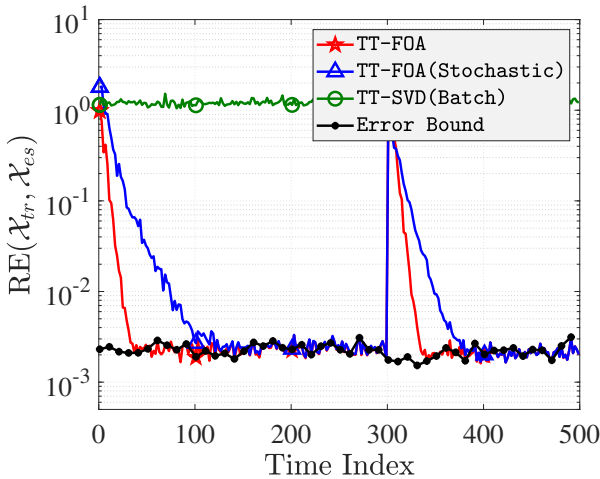
Figure 7.7: Performance of three TT decomposition algorithms in a time-varying scenario: The noise level $\epsilon = 10^{-1}$ and the time variance factor $\sigma = 10^{-4}$.

the same manner to the effect of the noise level, TT-FOA's estimation accuracy goes down when σ increases, but converges towards a steady state error. Fig. 7.7 shows a performance comparison among three TT decomposition algorithms when the value of the noise level ϵ and the time-varying factor σ are 10^{-1} and 10^{-4} respectively. The batch algorithm TT-SVD fails in this time-varying scenario, while TT-FOA and its stochastic version can
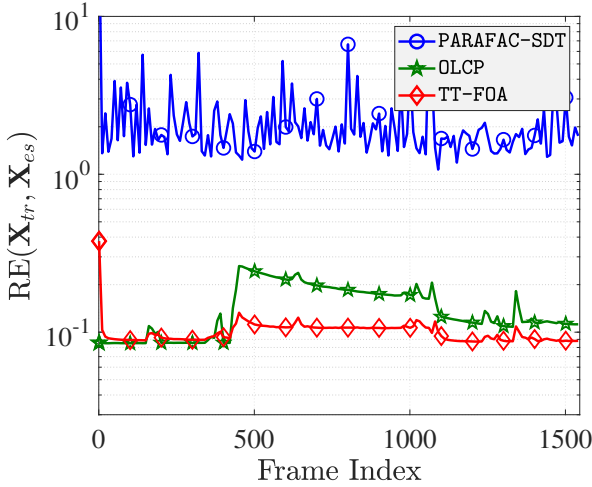
Figure 7.8: Track surveillance video: TT-rank r_{TT} = [15, 15] and CP-rank r_{CP} = 15.

track successfully the variation of the tensor along the time, which yields to an estimation accuracy very close to the error bound (i.e. steady state error). The result also indicates that the convergence rate of TT-FOA is faster than that of its stochastic version. This is probably because the convergence rate of the stochastic TT-FOA is limited by its noisy/stochastic approximation of the true gradient.

7.5.1.2 Real Data

In order to provide empirical evidences of applying TT-FOA to real data, we use a surveillance video sequence[2], and a functional MRI data[3]. The video data contains 1546 frames of size 128×160, while the fMRI data includes 20 abdominal scans of size $256 \times 256 \times 14$.

The first task is to track surveillance video. We compare TT-FOA against the two state-of-the-art adaptive CP tensor decompositions, including PARAFAC-SDT [211] and OLCP [175]. In order to apply these algorithms effectively, color video frames are converted into grayscale. The CP-rank and TT-rank are set at 15 and [15, 15] respectively. Moreover, the 100 first video frames are trained to obtain the good initialization for PARAFAC-SDT and OLCP. The results indicate that TT-FOA outperforms these adaptive CP decompositions, as shown in Fig. 7.8 and Fig. 7.9. In particular, PARAFAC-SDT fails to track video frame while OLCP achieves a worse estimation accuracy than our

---

[2]http://www.changedetection.net/
[3]https://github.com/colehawkins/

(a) Original Frame

(b) PARAFAC-SDT

(c) OLCP

(d) TT-FOA
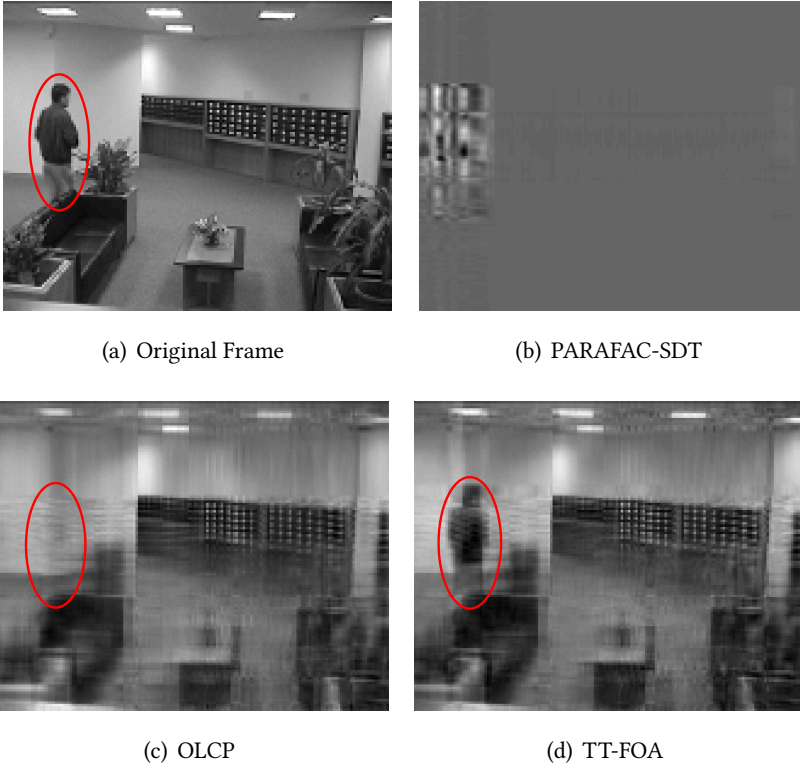
Figure 7.9: Reconstructed 1345**-th frame.**

algorithm.

The second task is to demonstrate the effect of TT-rank $\mathbf{r}_{TT}$ on the low-rank approximation of the fMRI tensor. The abdominal scans are seen as tensor slices in the online setting. Results of tracking the low-rank component of the last scan are shown in Fig. 7.10. The estimated low-rank fMRI scan deviates from its ground truth when the TT-rank decreases, and hence the relative error increases.

7.5.2 Performance of ATT

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.

(a) Grouth Truth

(b) RE = 0.077

(c) RE = 0.036

(d) RE = 0.007
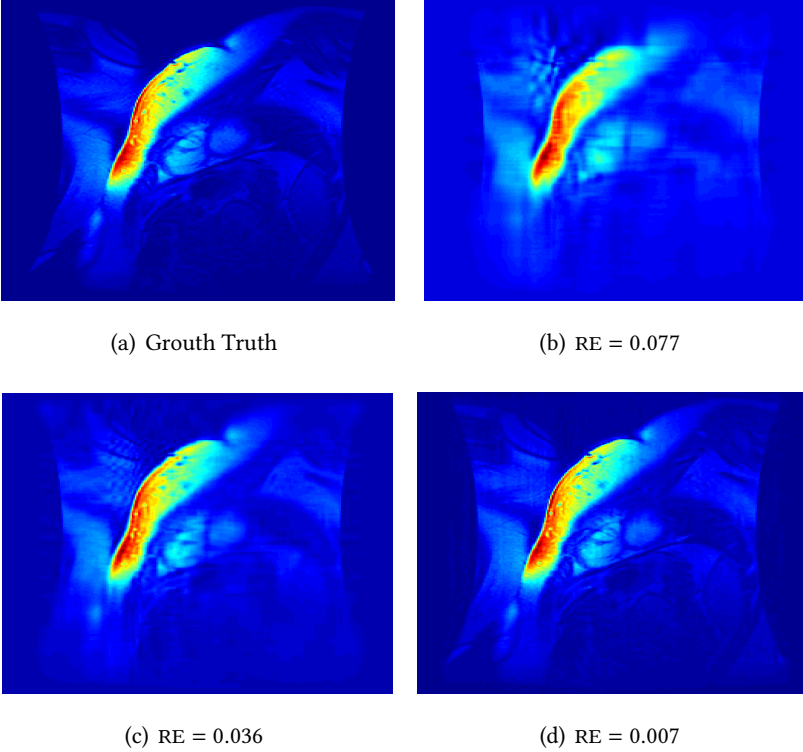
Figure 7.10: Effect of TT-rank on the low-rank approximation of fMRI scans: (a) original MRI scan, (b)-(d) low-rank approximation images for r_{TT} of $[10, 10]$, $[20, 20]$ and $[50, 50]$ respectively.

7.5.2.1 Experiment Setup

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

$$\boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{P}}_t \circledast \left(\boldsymbol{\mathcal{G}}_t^{(1)} \times_2^1 \boldsymbol{\mathcal{G}}_t^{(2)} \times_3^1 \boldsymbol{\mathcal{G}}_t^{(3)} \times_4^1 \mathbf{g}_t^{(4)} + \boldsymbol{\mathcal{N}}_t \right). \tag{7.63}$$

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 $1 - \omega_{\text{miss}}$, i.e., ω_{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 $\mathcal{N}(0, 1)$.
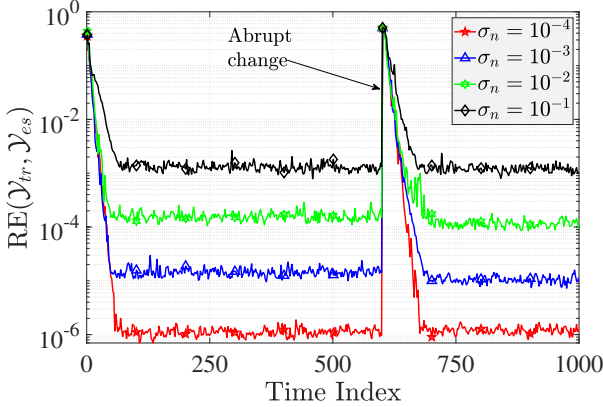
Figure 7.11: **Effect of the noise level σ_n on the tracking ability of ATT.**

We use the following relative error (RE) metric to evaluate the estimation accuracy:

$$\mathrm{RE}\big(\boldsymbol{\mathcal{Y}}_{tr}, \boldsymbol{\mathcal{Y}}_{es}\big) = \frac{\big\|\boldsymbol{\mathcal{Y}}_{tr} - \boldsymbol{\mathcal{Y}}_{es}\big\|_F}{\big\|\boldsymbol{\mathcal{Y}}_{tr}\big\|_F}, \tag{7.64}$$

where $\boldsymbol{\mathcal{Y}}_{tr}$ (resp. $\boldsymbol{\mathcal{Y}}_{es}$) refers to the true tensor (resp. reconstructed tensor).

7.5.2.2   Effect of the noise level σ_n

In this task, 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 ω_{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 $\epsilon = 1$ when $t = 600$ and $\varepsilon = 0$ otherwise) to investigate how fast ATT could converge. The result is illustrated in Fig. 7.11. We can see that the noise level σ_n does not affect the convergence rate of ATT but only its estimation error.

7.5.2.3   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. 7.12 illustrates the performance of ATT versus the value of ε. We can see that the

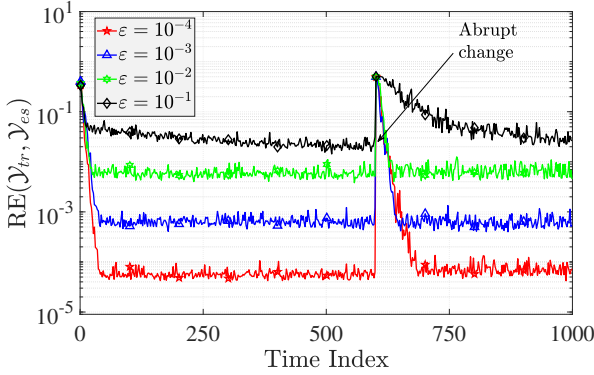**Figure 7.12: Effect of the time-varying factor ε on the tracking ability of ATT.**

estimation accuracy of ATT goes down when ε increases, but converges towards a steady-state error in the similar manner as in the previous case. Intuitively, the time-varying factor has an influence on the convergence rate of tracking algorithms. However, as shown in Fig. 7.12, the value of ε does not affect ATT's convergence rate. This "phenomenon" thus deserves further investigations.

7.5.2.4 Effect of the missing density ω_{miss}

Here, we measure the performance of ATT in the presence of different missing densities. Particularly, the value of ω_{miss} was chosen among $\{20\%, 40\%, 80\%\}$. We reused the same 4-order streaming tensor above with $\sigma_n = \varepsilon = 10^{-3}$. Fig. 7.13 shows that the number of missing entries in $\mathcal{X}_t$ has an impact on both convergence rate and estimation accuracy of ATT, i.e., the lower the value of ω_{miss} is, the better performance ATT achieves. However, even with 80% of missing data, ATT is still able to achieve relatively good performance.

7.5.2.5 Online video completion

Three real video sequences are used in this task, including "Lobby", "Highway", and "Hall". Their sizes are summarized in Table 7.1.

We compare ATT with other online tensor algorithms: TeCPSGD [106], ACP [30], and ATD [30]. To have a fair comparison, colour video frames were converted into grayscale ones. The CP-rank, Tucker-rank, and TT-rank were set to 10, [10, 10, 10], and [10, 10], respectively. The results in Table 7.1 (i.e., averaged relative errors) and Fig. 7.14 indicate that ATT provided a competitive video completion performance.
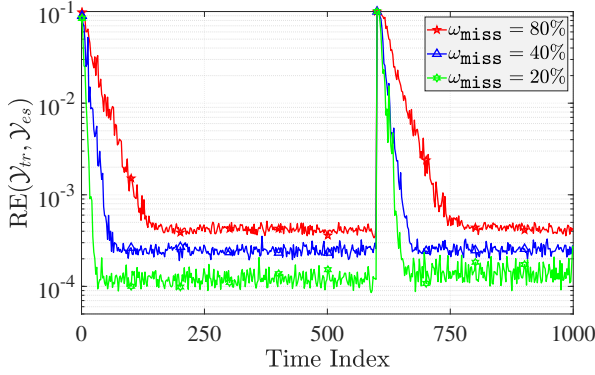
Figure 7.13: Effect of the missing density $\omega_{\texttt{miss}}$ on the tracking ability of ATT.
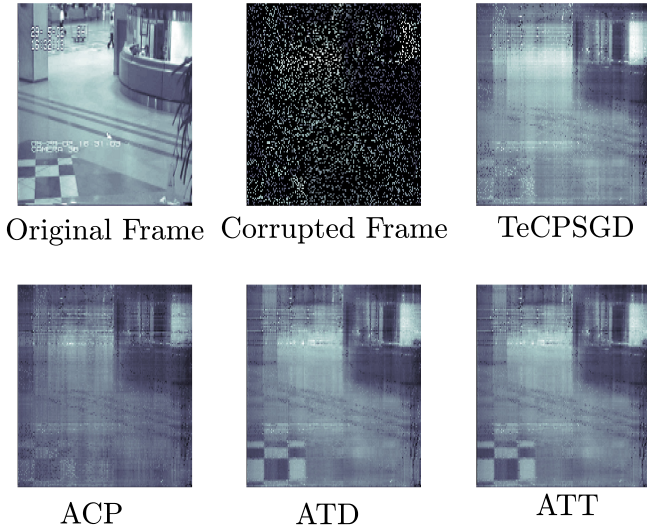
Figure 7.14: The 500**-th video frame of "Hall" data:** 80% **pixels are missing.**

7.5.3 Performance of ROBOT

We here evaluate the performance of ROBOT in terms of the following aspects: (i) impact of noise, (ii) its tracking ability in nonstationary environments, (iii) impact of missing observations, (iv) impact of outliers, and (v) its use for the problem of video background and foreground separation.

Table 7.1: Averaged relative error of adaptive tensor decompositions on incomplete video sequences.

Dataset	Size	Missing	Online Tensor Completion Methods			
			TeCPSGD	ACP	ATD	**ATT**
Hall	$174 \times 144 \times \times 3584$	20%	0.1351	0.1500	0.1366	**0.1264**
		40%	0.1412	0.1562	0.1370	**0.1272**
		80%	0.1547	0.1868	0.1472	**0.1336**
Lobby	$128 \times 160 \times \times 1546$	20%	0.1307	0.1320	0.1220	**0.1214**
		40%	0.1327	0.1375	0.1241	**0.1223**
		80%	0.1705	0.2142	0.1432	**0.1263**
Highway	$320 \times 240 \times \times 1700$	20%	0.2056	0.2204	0.1980	**0.1777**
		40%	0.2119	0.2206	0.2001	**0.1836**
		80%	0.2133	0.2481	0.2089	**0.2043**

7.5.3.1   Experiment Setup

We follow the problem formulation in Section II to simulate temporal slices $\{\boldsymbol{\mathcal{Y}}_t\}_{t \geq 1}$. In particular, $\boldsymbol{\mathcal{Y}}_t$ is randomly generated under the model

$$\boldsymbol{\mathcal{Y}}_t = \boldsymbol{\mathcal{P}}_t \circledast \left(\boldsymbol{\mathcal{L}}_t + \boldsymbol{\mathcal{O}}_t + \boldsymbol{\mathcal{N}}_t \right), \tag{7.65}$$

where $\boldsymbol{\mathcal{L}}_t = \boldsymbol{\mathcal{G}}_t^{(1)} \times_2^1 \boldsymbol{\mathcal{G}}_t^{(2)} \times_3^1 \boldsymbol{\mathcal{G}}_t^{(3)} \times_4^1 \mathbf{g}_t^{(4)}$. Here, $\boldsymbol{\mathcal{P}}_t \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times 1}$ is a binary mask tensor whose entries are obtained by a Bernoulli model with probability $1 - \omega_{miss}$ (i.e., ω_{miss} represents the missing density). $\boldsymbol{\mathcal{N}}_t$ is a Gaussian noise tensor whose entries are i.i.d. from $\mathcal{N}(0, \sigma_n^2)$. $\boldsymbol{\mathcal{O}}_t$ is a sparse tensor containing outliers whose amplitude is uniformly chosen in the interval $[0, \texttt{fac-outlier}]$ while their indices (locations) follow another Bernoulli model with probability $\omega_{outlier}$. $\boldsymbol{\mathcal{L}}_t$ is the low-rank component of $\boldsymbol{\mathcal{Y}}_t$ in which $\mathbf{g}_t^{(4)} \in \mathbb{R}^{r_3 \times 1}$ is a standard normal random vector. At time t, TT-cores are varied under the model $\boldsymbol{\mathcal{G}}_t^{(n)} = \boldsymbol{\mathcal{G}}_{t-1}^{(n)} + \varepsilon \boldsymbol{\mathcal{V}}_t^{(n)}$, where ε denotes the time-varying factor, $\boldsymbol{\mathcal{V}}_t^{(n)}$ shares the same size as $\boldsymbol{\mathcal{G}}_t^{(n)} \in \mathbb{R}^{r_{n-1} \times I_n \times r_n}$ and its entries are derived from $\mathcal{N}(0, 1)$. At $t = 0$, $\boldsymbol{\mathcal{G}}_0^{(n)}$ is initialized by a Gaussian distribution with zero mean and unit variance.
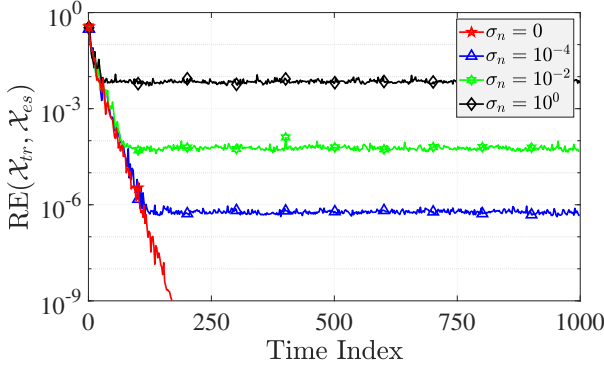
Figure 7.15: Effect of the noise level σ_n on the performance of ROBOT.

To evaluate the performance of ROBOT, we use the following relative error:

$$\mathrm{RE}\big(\mathcal{X}_{tr}, \mathcal{X}_{es}\big) = \frac{\big\|\mathcal{X}_{tr} - \mathcal{X}_{es}\big\|_F}{\big\|\mathcal{X}_{tr}\big\|_F}, \tag{7.66}$$

where $\mathcal{X}_{tr}$ (resp. $\mathcal{X}_{es}$) refers to the true low-rank component (resp. estimation).

7.5.3.2   Effect of the noise level σ_n

We change the value of σ_n and measure the estimation accuracy of ROBOT. We used a streaming tensor of size $10 \times 15 \times 20 \times 1000$ and rank $\mathbf{r}_{\mathrm{TT}} = [5, 5, 5]$. Parameters of the data model were set as: time-varying factor $\epsilon = 0$, missing density $\omega_{miss} = 0\%$, and outlier density $\omega_{outlier} = 0\%$ (i.e. outliers free observations). We fixed algorithmic parameters of ROBOT as follows: the forgetting factor $\beta = 0.5$ and two penalty parameters $\rho_1 = \rho_2 = 1$. The result is shown in Fig. 7.15. Clearly, the value of σ_n does not affect ROBOT's convergence rate but its relative error.

7.5.3.3   Effect of the time-varying factor ϵ

Next, we evaluate the performance of ROBOT in dynamic and nonstationary environments. We reused the streaming tensor above with 90% observations (i.e., $\omega_{miss} = 10\%$). The noise level σ_n was fixed at 10^{-3}. We set the outlier density and intensity to 10% and 1, respectively. The forgetting factor and two penalty parameters were kept as above. Also, an abrupt change was made at $t = 600$ to assess how fast ROBOT converges. Fig. 7.16 illustrates the effect of ϵ
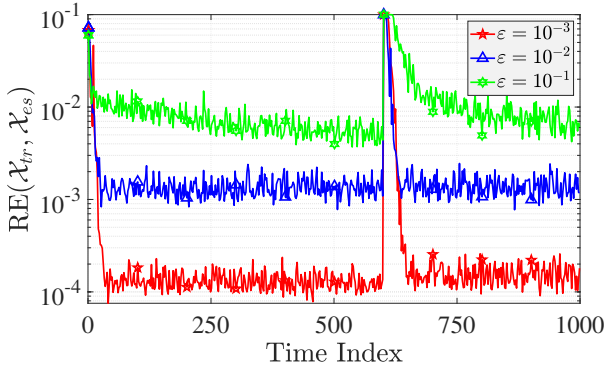
Figure 7.16: Effect of the varying factor ϵ on the performance of ROBOT.
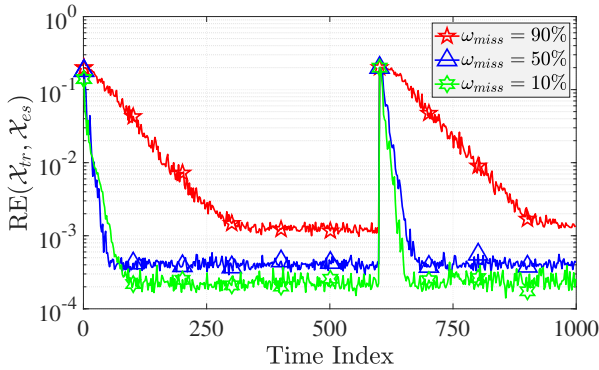
Figure 7.17: Effect of the missing density ω_{miss} on the tracking ability of ROBOT.

on the performance of ROBOT. We can see that the performance of ROBOT increases when ϵ decreases and converges towards a steady-state error.

7.5.3.4 Effect of the missing density ω_{miss}

We then investigate the tracking ability of ROBOT in the presence of missing data. The value of ω_{miss} was chosen among $\{10\%, 50\%, 90\%\}$. We kept all experimental parameters as above, except the time-varying factor ϵ which was set to 10^{-3}. We can see from Fig. 7.17 that both convergence rate and estimation accuracy of ROBOT are affected by the value of ω_{miss}. The lower $\omega_{\texttt{miss}}$ is, the better performance ROBOT achieves.

7.5.3.5 Effect of outliers

Here, we measure the robustness of ROBOT against sparse outliers. Most of experimental parameters were kept as in the previous tasks: $\omega_{miss} = 10\%$, $\beta = 0.5$, $\sigma_n = \epsilon = 10^{-3}$, and $\rho_1 = \rho_2 = 1$. We investigated the case when 30% entries were corrupted by outliers. Three levels of the outlier intensity `fac-outlier` were considered, including 0.1, 1, and 10 (resp. low, moderate, and strong effect). Fig. 7.18 indicates that ROBOT is capable of tensor tracking from incomplete observations corrupted by sparse outliers.
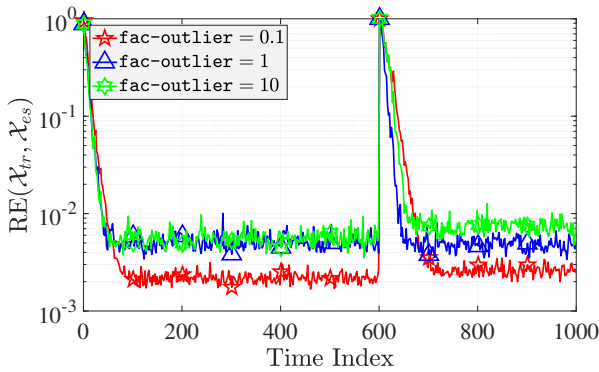
Figure 7.18: Effect of the outliers on the tracking ability of ROBOT.

7.5.3.6 Video background/foreground separation

In this task,[4] we used three video datasets, including "Lobby", "Highway", and "Hall". The dataset "Lobby" includes 1700 frames of size 144×176. There are 1700 frames of size 240×320 in the data "Highway, while "Hall" consists of 3584 frames whose size is 174×144. The performance of ROBOT was evaluated in comparison with two online background/foreground separation algorithms, including PETRELS-ADMM [25] and GRASTA [50]. The subspace rank and TT-rank were set to 10 and $[10, 10]$, respectively. The result from Fig. 7.19 indicates that ROBOT is able to detect moving objects in real surveillance video sequences with reasonable performance.

7.6 Conclusions

In this chapter, we have considered the problem of tensor tracking under the tensor-train format. Three novel adaptive tensor-train decomposition al-

---

[4]Here, the foreground plays the role of outliers and its separation from the background is based on the proposed detection procedure.

Figure 7.19: Background and foreground separation. From bottom to top row: Highway, Hall, and Lobby. From left to right column: Original video frame, PETRELS-ADMM, GRASTA, and ROBOT.

gorithms are proposed for factorizing streaming tensors, including TT-FOA, ATT, and ROBOT. Each algorithm is specifically designed for dealing with a specific task. In particular, the former algorithm TT-FOA and its stochastic variant have the capability to track the tensor-train representation of streaming tensors from noisy and high-dimensional data with high accuracy, even when they come from time-dependent observations. By utilizing the recursive least-squares method in adaptive filtering, the second algorithm 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 latter algorithm ROBOT – which is a robust version of ATT – is fully capable of tracking the underlying low-rank component of incomplete streaming tensors corrupted by sparse outliers in nonstationary environments. All three algorithms are fast, effective, and requires low computational complexity and memory storage. To the best of our knowledge, they are the first of their kind that have the potential to handle streaming tensors under the tensor-train format.

Conclusion and Outlook | 8

8.1 Conclusions

In this thesis, we have presented several contributions to the problem of tracking the low-rank approximation of big data streams over time.

For Subspace Tracking

- We provided a survey on recent robust subspace tracking (RLS) algorithms to fill the gap in the literature particularly addressing non-Gaussian noises (i.e., outliers, impulsive noise, and colored noise) and sparse constraints. In the context of missing data and outliers, we reviewed four main classes of RST algorithms, including Grassmannian, recursive least-squares (RLS), recursive projected compressive sensing (ReProCS), and adaptive projected subgradient method (APSM). When the data streams are corrupted by impulsive noises, we indicated that most of state-of-the-art subspace tracking algorithms are based on improving the well-known PAST algorithm, together with weighted RLS and adaptive Kalman filtering. Next, we outlined two main approaches to deal with subspace tracking in the presence of colored noises, including instrumental variable-based and oblique projections. Finally, a short review on sparse subspace tracking algorithms was presented.

- We proposed a probable adaptive algorithm called PETRELS-ADMM for tracking the underlying subspace from incomplete observations corrupted by sparse outliers. The proposed algorithm contains two main stages: outlier rejection and subspace estimation. In particular, outliers residing in the measurement data are detected and removed by our ADMM solver in an effective way. Next, we proposed an improved version of PETRELS, namely iPETRELS. It is observed that PETRELS is ineffective when the fraction of missing data is too large. We thus added a regularization of the $\ell_{2,\infty}$-norm, which aims to control the maximum ℓ_2-norm of rows in the subspace matrix, in the objective function to avoid such performance loss. Moreover, we also introduced an adaptive step size to speed up the convergence rate as well as enhance the sub-

space estimation accuracy. Furthermore, we successfully established a theoretical convergence which guarantees that the solutions generated by PETRELS-ADMM will converge to a stationary point asymptotically.

- We proposed a novel adaptive algorithm called OPIT for the sparse subspace tracking (SST) problem. OPIT takes both advantages of power iteration and thresholding methods, and hence offers several appealing features over the state-of-the-art tracking algorithms. First, OPIT belongs to the class of power methods, and thus its convergence rate is highly competitive compared to other SST algorithms, especially in the high SNR regime. Different from the existing two-stage SST algorithms, OPIT has ability to track the sparse principal subspace with high accuracy in both the classical regime and the HDLSS regime. In addition, OPIT is flexible and very adaptable for different scenarios. For example, we can adjust its procedure for dealing with multiple incoming data streams. Also, it is easy to introduce regularization parameters into OPIT in order to regularize its performance in non-standard environments. Moreover, we can recast its update rule into a column-wise update. Thanks to the deflation transformation, we derived a fast variant of OPIT called OPITd with lower complexity of both computation and memory storage. This variant is fast and useful for tracking high-dimension and large-scale data streams residing in a low-dimensional space. Together with PETRELS-ADMM, OPIT belongs to the class of provable subspace tracking algorithms in which its convergence is guaranteed. Under certain conditions, OPIT can achieve an ϵ-relative-error approximation with high probability when the number of observations is large enough.

For Tensor Tracking

- We provided a comprehensive survey on the state-of-the-art tensor tracking algorithms. It begins with basic coverage of five common tensor decompositions and their main features, including CP, Tucker, BTD, tensor-train, and t-SVD. Two kinds of streaming models were introduced to represent streaming tensors: single-aspect and multi-aspect. Next, we reviewed four main classes of online CP algorithms: subspace-based, block-coordinate descent, Bayesian inference, and multi-aspect streaming CP decomposition. Under the Tucker format, we categorized the current single-aspect tensor tracking algorithms into two main classes: online tensor dictionary learning and tensor subspace tracking. Multi-aspect streaming Tucker decomposition algorithms were also surveyed. Moreover, we provided a brief survey on other online techniques for tracking tensors under tensor-train, t-SVD, and BTD formats.

- We proposed three efficient adaptive algorithms for tracking the low-rank component of streaming tensors over time. Under the CP format, we developed a novel adaptive CP algorithm called ACP for tracking high-order incomplete streaming tensors. ACP is fast and requires a low computational complexity and memory storage, thanks to the alternative minimization and randomized sketching. Under the Tucker format, we proposed the second algorithm, namely adaptive Tucker decomposition (ATD), more flexible than ACP, for the problem of tensor tracking. ATD exhibits competitive performance in terms of both estimation accuracy and computational complexity. Third, we introduced a robust version of ACP called RACP for the problem of tensor tracking in the presence of both missing data and outliers. In particular, RACP aims to learn the low-rank component of streaming tensors in an online fashion as well as offering robustness against gross data corruptions. More importantly, we proved that ACP, ATD, and RACP are provable algorithms with convergence guarantee.

- We developed three new methods for the problem of tensor tracking under the tensor-train (TT) format. The first method called TT-FOA is capable of tracking the low-rank components of high-order tensors from noisy and high-dimensional data with high accuracy, even when they come from time-dependent observations. The second method called ATT is particularly designed for handling incomplete streaming tensors. ATT is scalable, effective, and adept at estimating low TT-rank component of streaming tensors. Besides, ATT can support parallel and distributed computing. To deal with sparse outliers, we proposed the so-called ROBOT which stands for ROBust Online Tensor-Train decomposition. Technically, ROBOT has the ability to tracking streaming tensors from imperfect streams (i.e., due to noise, outliers, and missing data) as well as tracking their time variation in dynamic environments.

8.2 Research Challenges, Open Problems, and Future Directions

In this section, we present several research challenges and open problems that should be considered for the development of tensor tracking problems in the future. These problems also cover the subspace tracking problem as it is a special case of tensor tracking. They are data imperfection and corruption; rank revealing and tracking; efficient and scalable tensor tracking; and other aspects such as theoretical analysis, symbolic data, and tracking under some less common tensor formats. Possible solutions for these challenges are also

discussed.

8.2.1   Data Imperfection and Corruption

Dealing with data imperfection and corruption has been a critical issue in many applications and tracking problems in particular [367]. We here present two main types of imperfect data that either remain unsolved or are still challenging for tensor tracking: (i) non-Gaussian and colored noises; (ii) outliers and missing data.

Non-Gaussian and Colored Noises

Most of the existing tensor tracking algorithms were proposed under the additive white Gaussian noise assumption. This assumption however does not always hold in practice. For example, impulsive noises (e.g., burst, alpha-stable, and spherically invariant random variable noise), which are introduced by human activities and natural sources, are one of the most common non-Gaussian noises that often appear in tracking applications such as direction of arrivals [368], OFDM systems [369] and adaptive system identification [370]. This type of noise can significantly impact the tracking ability of estimators and it requires specific treatments [26]. In parallel, colored noises that indicate types of noise that are correlated in space and/or time may reduce the performance of tracking algorithms [371]. Accordingly, standard tracking algorithms may be less effective in estimation accuracy in the presence of these noises. They need to be readapted or redesigned for more robustness.

    To the best of our knowledge, we are not aware of any tensor tracking algorithm capable of handling such noises in the literature. Some potential approaches have been successfully demonstrated in subspace tracking problems (i.e., tracking tensors of order 2), see [26] for a brief survey. In particular, adaptive Kalman filtering and weighted RLS approaches can be adopted for dealing with impulsive noises. Oblique projection and instrumental variable-based techniques can handle colored noises. Therefore, it is desirable to extend these approaches from subspace tracking to tensor tracking.

Outliers and Missing Data

They are now becoming more and more ubiquitous in modern datasets. Outliers are data points that appear to be inconsistent with or exhibit abnormal behaviour different from others. Missing observations are often encountered during the data acquisition and collection. Both outliers and missing data can cause several issues (e.g., they introduce bias in estimation) for knowledge discovery from data in general and data streams in particular [6]. Accordingly, dealing with them is an essential task in the analysis of corrupted datasets

which has been still a hot topic in data mining for decades. In general, handling such corruptions involves removing/ignoring them after detection or replacing them with alternative values.

There exist few tensor tracking algorithms robust to sparse outliers in the literature. Under the CP format, SOFIA [222] applies the robust Holt-Winters forecasting model using a pre-cleaning mechanism to identify and down-weight outliers. RACP [27] introduces a ℓ_1-norm penalty to promote the sparsity on outliers and then uses an ADMM solver to estimate them. Under the Tucker format, ORLTM [263], OLRTR [264], and D-L1-Tucker [254] are able to deal with sparse outliers. Both ORLTM and OLRTR propose to regularize the main objective function with a ℓ_1-norm regularization. Meanwhile, D-L1-Tucker adopts a threshold-based method to detect outliers. Except for RACP, most of the mentioned algorithms above are not designed for dealing with missing data. In parallel, most of the existing online tensor completion and tracking are sensitive to outliers, such as TeCPSGD [106], OLSTEC [176], and ACP [30]. Accordingly, there are plenty of opportunities for us to develop robust tensor tracking from incomplete observations as it is still in its early stage.

8.2.2   Rank Revealing and Tracking

Most of the state-of-the-art tensor tracking algorithms suppose that the tensor rank (e.g., CP, Tucker, TT, or tubal rank) is given as prior information. In practice, it is however a difficult assumption due to the facts that: (i) the tensor rank may change over time and (ii) a good rank determination at the initialization stage is not always guaranteed when the number of training samples is limited and (iii) the exact rank determination may be intractable (e.g., CP rank is NP-hard [195]). Therefore, it is essential to develop tracking algorithms that are capable of revealing the rank over time.

In the literature, there have been many heuristic methods developed for the problem of tensor rank estimation. Most of them adopt the Bayesian approach to infer the tensor rank from data, such as [372–374]. Theoretically, Bayesian inference offers a good recipe for the tensor rank estimation as we can integrate the low-rank promoting prior as well as the tensor rank into the learning framework. Another possible approach to determine the tensor rank is to use neural networks (NNs), such as [375–377]. Since the rank can be considered as one type of data feature, NNs which can extract hidden features within data can be used to solve the tensor rank determination. Although these methods often require the tensor data to be fully observed, it is possible to readapt or modify them such that their variant are able to handle tensors in an online fashion. For example, we can adopt online Bayesian inference or online learning algorithms for training NNs.

8.2.3 Efficient and Scalable Tensor Tracking

Chapter 5 indicates that most of the existing tensor tracking algorithms are of high complexity. When we deal with large-scale and high-multidimensional streams, they may become less efficient. Thus, it is necessary to develop efficient and scalable tracking techniques of low cost w.r.t. both computational complexity and memory storage. In what follows, we present three potential approaches which are theoretically capable of accelerating the tracking process, namely (a) randomized sketching, (b) parallel and distributed computing, and (c) neural networks-based methods.

Randomized Sketching

It is very well-known that randomized methods can reduce the computational cost of their counterparts while still achieving reasonable estimation [323]. Accordingly, many attempts have been made to take their advantages in computation for tensor decomposition in the literature, we refer the readers to [191] for a good overview. Among them, there are a few online algorithms utilizing successfully randomized techniques to speed up the tracking process, such as [30, 33, 218, 378]. Particularly, these algorithms involve solving several overdetermined least-squares (LS) problems. Thanks to the CP and Tucker structures, they use random sampling to build the sampled Khatri-Rao and Kronecker products, and then, recast the original LS problems into randomized ones. Solving the new LS problems can save a lot of computational complexity. Other randomized techniques (e.g., random projections and count sketch) with other tensor formats have not yet been investigated for tensor tracking and they deserve next investigations in the future.

Parallel and Distributed Computing

The second approach is to develop parallel and distributed computing frameworks for streaming tensor decomposition. It stems from the fact that we can leverage several computational resources to facilitate the tracking process. Moreover, computing systems in a parallel and distributed environment can offer more reliability than their counterparts in a central one as they can avoid the single point of failure which is a fundamental mistake from flaws in the implementation or design of a system. Besides, another appealing advantage of this computing is the scaling up-and-out process in which we can add and/or replace computational resources to the system. We refer the readers to [379] for a good reference.

In the tensor literature, there are several parallel and distributed systems for processing large-scale tensors. We can list here some efficient tools for:

(a) distributed CP decomposition (e.g., DFacTo [380], SPLATT [381]), (b) distributed Tucker decomposition (e.g., DHOSVD [246], SGD-Tucker [382]), and (c) distributed TT decomposition (e.g., ADTT [268], ATTAC [383]), etc. These tools mainly distribute the unfolding matrices or sub-tensors among several clusters and integrate their low-rank tensor approximations to find the overall low-rank approximation of the underlying tensor. However, most of the existing distributed tensor decompositions are not suitable for handling streaming data. Therefore, it is of great interest to develop practical distributed systems for tracking tensors from data streams.

Neural Networks-based Methods

Another potential approach is to incorporate neural networks (NNs) into tensor factorization to benefit from their significant advances in computational power. On the one hand, the connection between TDs and NNs has been established in some studies, such as [384–386]. For example, Cohen *et al.* in [384] showed that the convolutional NNs with ReLU activation and max/average pooling can be represented by tensor decomposition models. Wang *et al.* in [386] introduced two NN models for finding the low-tubal-rank approximation of three-order tensors. Accordingly, NN tools can be used to model and learn high-order interactions for tensors, and hence, for tensor factorization and tracking. On the other hand, NNs can directly map data streams (temporal slices) as input to the approximation result as output by applying some online learning techniques. In the literature of machine learning, there exist several kinds of learning capable of dealing with data streams, such as incremental learning, lifelong learning, and online continual learning, to name a few. They can be specifically adapted for tensor tracking.

8.2.4 Others

Next, we present some other issues and problems which also deserve future investigations.

Provable Tensor Tracking

Although the existing tensor tracking methods can provide competitive performance w.r.t. estimation accuracy and/or convergence rate in practice, most of them lack performance guarantees. The gap between practical uses/implementations and theoretical results in tensor tracking may be caused by the fact that most tensor problems are NP-hard [195], e.g., the best rank-1 tensor approximation is NP-hard even when all observations (temporal slices) are fully observed. Despite several difficulties, there are still attempts to bridge

the gap in the literature. Under certain conditions (e.g., the underlying low-rank model remains unchanged over time), some studies established successfully theoretical results to analyse the convergence behavior of their methods, such as [27, 30, 176, 219, 251]. These initial results encourage us to investigate deeper theory aspects in tensor tracking, such as time variation, asymptotic convergence, and non-asymptotic convergence in low-sample-size settings.

Symbolic Tensor Tracking

In some applications, data may no longer be represented by single (certain) values, but need to be formatted or grouped within sets, intervals, histograms, etc. It leads to the so-called symbolic data analysis (SDA) paradigm in data mining and statistics to deal with such data [387]. In SDA, several new variables types and processing tools have been introduced to represent and analyse symbolic data, such as interval-valued, histogram-valued, and categorical modal variables, to name a few. The readers are referred to [387] for a good survey on SDA. In the tensor literature, Mauro *et al.* in [388] proposed for the first time a symbolic tensor decomposition for factorizing interval-valued tensors under the tensor-train format. Specifically, the authors extended a set of tools aiming to handle interval-valued matrices for high-order tensors and introduced efficient decomposition and reconstruction strategies. As the symbolic tensor decomposition is in its very early stage of development in both batch and online settings, there are a lot of aspects that need to be investigated in the future.

Tensor Tracking under BTD, t-SVD, Tensor Network formats, and other Variants

As reviewed in the sections above, most of the state-of-the-art tensor tracking algorithms are proposed for streaming CP and Tucker decompositions. Despite great success in the batch setting, BTD, t-SVD, and tensor networks (e.g., tensor-train, tensor chain, and tensor ring) have not attracted much attention in real-time stream processing until recently. Thus, developing online methods for tracking tensors under these tensor formats and their variants is essential advantage from their advantage in representing large-scale tensors as well as fulfil the gap between the two most common tensor formats and others.

Résumé de la Thèse
A

A.1   Traitement de Flux de Données Volumineuses

Le traitement de flux a récemment attiré beaucoup d'attention de la part des universités et de l'industrie en raison du fait que des flux de données massifs ont été de plus en plus collectés au fil des ans et qu'ils peuvent être exploités intelligemment pour découvrir de nouvelles idées et des informations précieuses [1–3]. Par exemple, nous vivons à l'ère de l'Internet des objets où un grand nombre de dispositifs de détection ont été installés et développés. Ces appareils ont la capacité de collecter, gérer et transmettre des données via des réseaux IoT en temps réel. En conséquence, le traitement de flux est nécessaire pour récupérer des informations importantes à partir de ces données IoT en quelques secondes ou même plus rapidement pour faciliter la prise de décision en temps réel [4].

Dans de nombreuses applications en ligne modernes, les flux de données ont trois caractéristiques en «V»: *Volume*, *Vitesse*, et *Variété*. Comme ils sont générés en continu, leur volume croît significativement dans le temps et éventuellement jusqu'à l'infini. Ainsi, l'une des caractéristiques les plus remarquables des données en continu est qu'il s'agit de séquences illimitées d'échantillons de données. *Vitesse* fait référence au taux d'arrivée de données à grande vitesse et au traitement en temps réel. Les données collectées à partir des interactions des utilisateurs sur les réseaux sociaux (par ex. Facebook, Instagram et Twitter) sont, par exemple, à très grande vitesse. *Variété* implique l'adéquation, la crédibilité et la fiabilité des flux de données. Plus précisément, cette caractéristique concerne le biais, le bruit, l'incertitude, l'incomplétude et l'anormalité des données. Outre les trois « V », les données en continu ont d'autres caractéristiques distinctives, notamment la sensibilité/variation temporelle (alias dérive de concept), l'hétérogénéité (différentes sources avec une diversité de types de données), une propriété volatile et non reproductible, etc [2, 3, 5, 6]. Ces caractéristiques entraînent plusieurs exigences inhérentes et des problèmes de calcul pour le traitement des flux, tels que:

- *Faible Latence*: Les procédés et systèmes de flux doivent acquérir, gérer

et traiter efficacement des flux de données sans introduire de retards supplémentaires.

- *Faible Complexité Spatiale*: Les procédés et systèmes de flux doivent avoir la capacité de fonctionner en ligne avec des ressources de mémoire limitées.

- *Évolutivité*: Comme les données de streaming augmentent normalement en taille beaucoup plus rapidement que les ressources de calcul, le traitement de flux nécessite des méthodes et des systèmes évolutifs.

- *Variation Temporelle*: Comme les données en continu peuvent évoluer avec le temps, les méthodes et les systèmes de flux doivent être capables de suivre leur variation dans le temps.

- *Robustesse*: Dans de nombreux cas, les données de flux sont imparfaites et peu fiables, de sorte que les méthodes et les systèmes de flux devraient avoir le potentiel d'estimer et de calculer les réponses à partir d'observations corrompues.

Cependant, ce sont également des avantages potentiels du traitement par flux par rapport au traitement par lots. Nous renvoyons les lecteurs au Tableau. A.1 pour une brève comparaison entre les deux types de traitement.

Table A.1: Principales différences entre le traitement par lots et le traitement des flux

Caractéristique	Traitement par lots	Traitement des flux
Input	Grands lots/morceaux de données	Flux de données (continus)
Taille des données	Connu et fini	Inconnu et/ou infini
Type de données	Statique	Dynamique/variant dans le temps
Traitement	Traiter toutes les données à la fois Traitement en plusieurs passes	Traiter les flux de données en temps réel Processus en un ou deux passages
Réponse	Fournir après l'achèvement	Fournir immédiatement
Hardware	Nécessite beaucoup de stockage Nécessite beaucoup de traitement	Nécessite beaucoup moins ou pas de stockage Nécessite moins de ressources de traitement
Temps	Prend plus de temps, latences en minutes à heures	Prenez quelques secondes ou plus vite

Dans ce travail, nous nous concentrons principalement sur les méthodes de flux capables de suivre l'approximation de rang inférieur (LRA) des flux de données volumineuses au fil du temps. Techniquement, l'objectif principal de
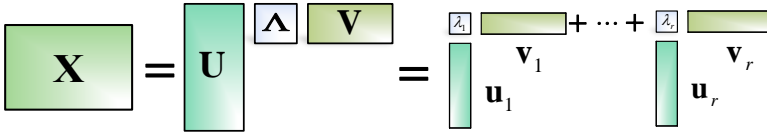
Figure A.1: SVD d'une matrice X.

la LRA est d'approximer les données de grande dimension par une représentation de faible dimension plus compacte avec une perte d'informations limitée [7]. Par conséquent, trouver la LRA est une tâche fondamentale et essentielle pour l'exploration de données en général et l'analyse de données en continu en particulier. Nous introduisons l'une des techniques d'algèbre linéaire les plus connues pour trouver le LRA des matrices dans la configuration par lots, la décomposition en valeurs singulières (SVD), puis décrivons sa connexion à certains types courants de décomposition tensorielle (TD). Ensuite, nous présentons leurs variantes en ligne (adaptatives) pour traiter les flux de données issus de l'observation unidimensionnelle. (i.e., SVD → sous-espace) et observations multidimensionnelles (i.e., TD → suivi tensoriel).

A.1.1 Approximation de Rang Inférieur: Du SVD au Décomposition du Tenseur

Il est bien connu que SVD est l'une des techniques d'algèbre linéaire les plus puissantes et les plus utilisées avec un certain nombre d'applications dans divers domaines [8, 9]. En particulier, décomposition SVD d'une matrice $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$ ima rang r est

$$\mathbf{X} \overset{\text{SVD}}{=} \underbrace{\left[\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r\right]}_{\mathbf{U}} \underbrace{\begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_r \end{bmatrix}}_{\mathbf{\Lambda}} \underbrace{\begin{bmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \vdots \\ \mathbf{v}_r^\top \end{bmatrix}}_{\mathbf{V}^\top} = \sum_{i=1}^{r} \lambda_i \mathbf{u}_i \mathbf{v}_i^\top, \qquad \text{(A.1)}$$

où $\mathbf{U} \in \mathbb{R}^{I_1 \times r}$ et $\mathbf{V} \in \mathbb{R}^{I_2 \times r}$ sont des matrices unitaires; $\mathbf{\Lambda} \in \mathbb{R}^{r \times r}$ est une matrice diagonale dont les valeurs diagonales sont positives, c.-à-d. $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > 0$, voir Fig. A.1 pour une illustration. Pour le problème d'approximation de rang inférieur dans la configuration par lots, le théorème suivant indique que SVD peut donner le meilleur LRA pour n'importe quelle matrice X.

Theorem 5 (Eckart-Young-Mirsky Theorem [9]) *Dénoter par* $\mathbf{X} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^\top$ *décomposition SVD d'une matrice* $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$. *Si* $k \leq rank(\mathbf{X})$ *et* $\mathbf{X}_k = \sum_{i=1}^{k} \lambda_i \mathbf{u}_i \mathbf{v}_i^\top$, *donc*

$$\min_{\substack{\mathbf{A} \in \mathbb{R}^{I_1 \times I_2} \\ rank(\mathbf{A}) \leq k}} \|\mathbf{X} - \mathbf{A}\| = \|\mathbf{X} - \mathbf{X}_k\|, \tag{A.2}$$

par rapport à la norme spectrale et à la norme de Frobenius.

Merci au Théorème 5, la meilleure approximation rang-k de $\mathbf{X}$ peut être obtenue en appliquant la procédure suivante:

- *Étape 1*: Calculer $\mathbf{X} \overset{\text{SVD}}{=} \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^\top$, où $\mathbf{U} \in \mathbb{R}^{I_1 \times I_1}$ et $\mathbf{V} \in \mathbb{R}^{I_2 \times I_2}$ sont des matrices unitaires, et la matrice diagonale $\boldsymbol{\Lambda} \in \mathbb{R}^{I_1 \times I_2}$ contient des entrées diagonales positives dans l'ordre décroissant.

- *Étape 2*: Sélectionnez les premiers k vecteurs singuliers parmi $\mathbf{U}$ et $\mathbf{V}$ pour former les matrices suivantes $\mathbf{U}_k = \mathbf{U}(:, 1:k)$ et $\mathbf{V}_k = \mathbf{V}(:, 1:k)$.

- *Étape 3*: Sélectionnez les k valeurs singulières les plus fortes dans $\boldsymbol{\Lambda}$ pour former: $\boldsymbol{\Lambda}_k = \boldsymbol{\Lambda}(1:k, 1:k)$.

- *Étape 4*: Dérivez la meilleure approximation rang-k de $\mathbf{X}$ à partir de: $\mathbf{X}_k = \mathbf{U}_k \boldsymbol{\Lambda}_k \mathbf{V}_k^\top$.

Lorsqu'il s'agit de tenseurs (aka, tableaux multidimensionnels), plusieurs extensions multivoies du SVD ont été développées pour la décomposition tensorielle (TD) dans la littérature [10–13]. Les cinq types courants de TD sont CP/PARFAC [14], Tucker/HOSVD [15], tensor-train/network [16], t-SVD [17], et décomposition en termes de blocs (BTD) [18], voir Fig. A.2 pour des illustrations. Plus précisément, ils visent à factoriser un tenseur en un ensemble de composants de base (par exemple, des vecteurs, des matrices ou des tenseurs plus simples) et offrent donc de bonnes approximations de tenseur de rang inférieur. Dans ce qui suit, nous décrivons leur connexion à SVD et renvoyons les lecteurs au Chapitre 5 pour plus de détails sur leurs principales caractéristiques, propriétés et algorithmes.

Décomposition CP/PARAFAC: Semblable à SVD qui représente $\mathbf{X}$ par une somme de matrices de rang-1 (c.-à-d. $\lambda_i \mathbf{u}_i \mathbf{v}_i^\top$), la décomposition CP factorise également un tenseur $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ en termes de rang-1:

$$\boldsymbol{\mathcal{X}} \overset{\text{CP}}{=} \sum_{i=1}^{r} \lambda_i \underbrace{\mathbf{u}_i^{(1)} \circ \mathbf{u}_i^{(2)} \circ \cdots \circ \mathbf{u}_i^{(N)}}_{\text{rank-1 term}}, \tag{A.3}$$
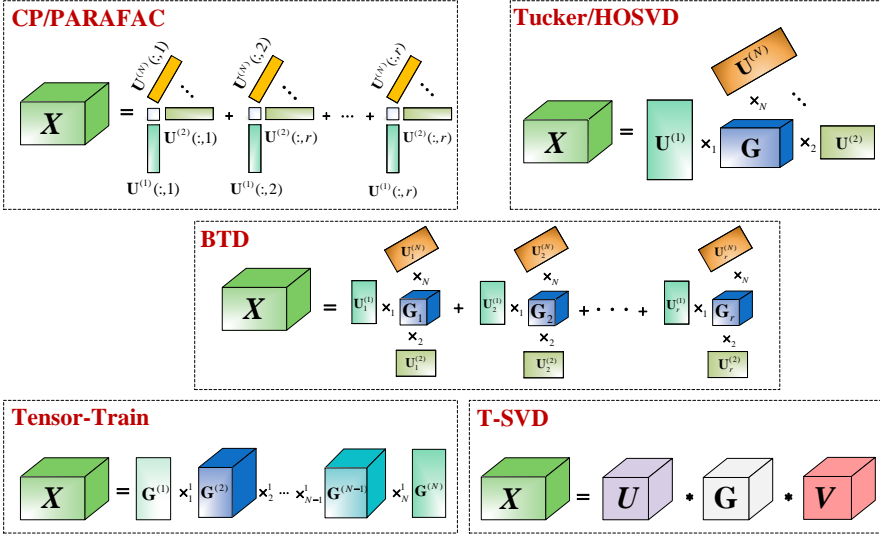
Figure A.2: Multiway extensions of SVD to high-order tensors: CP/-PARAFAC, Tucker, BTD, tensor-train, and t-SVD.

où $\mathbf{u}_i^{(n)} \in \mathbb{R}^{I_n \times 1}$ avec $1 \leq n \leq N$ joue le même rôle que les vecteurs singuliers de $\mathbf{U}$ et $\mathbf{V}$ dans le modèle SVD (A.1) (notez que $\mathbf{u}_i \mathbf{v}_i^\top = \mathbf{u}_i \circ \mathbf{v}_i$) [14]. La matrice $\mathbf{U}^{(n)} = \left[\mathbf{u}_1^{(n)}, \mathbf{u}_2^{(n)}, \ldots, \mathbf{u}_r^{(n)} \right]$ est la n-ième facteur CP de $\mathcal{X}$ et il n'est pas nécessaire qu'il soit orthogonal. Suivant la définition générale du rang de la matrice, le plus petit entier r satisfaisant (A.3) est appelé le rang du tenseur (CP) de $\mathcal{X}$. Dans certaines conditions, la décomposition CP est essentiellement unique jusqu'à une permutation et une échelle qui est une propriété utile dans de nombreuses applications.

Décomposition Tucker/HOSVD: En dehors de la forme classique (A.1), on peut exprimer la SVD de $\mathbf{X}$ comme suit

$$\mathbf{X} \overset{\text{SVD}}{=} \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^\top = \underbrace{\boldsymbol{\Lambda}}_{\text{core}} \underbrace{\times_1 \mathbf{U} \times_2 \mathbf{V}}_{\text{2 factors}}. \tag{A.4}$$

En conséquence, une extension multidirectionnelle directe de (A.4) aux tenseurs d'ordre élevé peut être donnée par

$$\mathcal{X} \overset{\text{Tucker}}{=} \underbrace{\mathcal{G}}_{\text{core}} \underbrace{\times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_N \mathbf{U}^{(N)}}_{N \text{ factors}}, \tag{A.5}$$

où le noyau $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times \cdots r_N}$ est un tenseur de taille inférieure à $\mathcal{X}$ (i.e., $r_n \leq I_n \; \forall n$) et N facteurs tensoriels $\{\mathbf{U}^{(n)}\}_{n=1}^N$, $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times r_n}$ sont des matrices orthogonales. Le modèle de représentation (A.5) est considéré comme le

format SVD d'ordre élevé (HOSVD) ou Tucker [15]. Contrairement au SVD et au CP, Tucker/HOSVD n'est pas unique en général. Cependant, comme le sous-espace couvrant $\mathbf{U}^{(n)}$ est physiquement unique, son objectif principal est de trouver les sous-espaces principaux des facteurs tensoriels [11].

Décomposition BTD: BTD factorise $\mathcal{X}$ en plusieurs blocs de rang multilinéaire faible au lieu de termes rank-1

$$\mathcal{X} \overset{\text{BTD}}{=} \sum_{i=1}^{r} \underbrace{\mathcal{G}_i \times_1 \mathbf{U}_i^{(1)} \times_2 \mathbf{U}_i^{(2)} \times_3 \cdots \times_N \mathbf{U}_i^{(N)}}_{\text{low multilinear-rank term}} . \tag{A.6}$$

Le BTD peut être considéré comme une unification et une généralisation des deux décompositions CP et Tucker bien connues. Plus précisément, lorsque $\{\mathcal{G}_i\}_{i=1}^{r}$ sont des tenseurs diagonaux, BTD se résume à la décomposition CP. Il a la forme d'une décomposition de Tucker lorsqu'un seul terme de bloc (c'est-à-dire $r = 1$) est considéré. De plus, plusieurs fonctionnalités attrayantes du BTD sont héritées de CP et de Tucker, telles que le calcul stable de Tucker, l'identification et l'unicité de CP [18]. En parallèle, il convient de rappeler une remarque dans [18] selon laquelle "le rang d'un tenseur d'ordre supérieur est en fait une combinaison des deux aspects : il faut préciser le nombre de blocs et leur taille". Cela signifie que BTD fournit une approche unifiée pour généraliser le concept de rang matriciel aux tenseurs.

Décomposition Tensor-Train: Avec (A.1) et (A.4), nous pouvons écrire la SVD de $\mathbf{X}$ comme

$$\mathbf{X}(i_1, i_2) \overset{\text{SVD}}{=} \sum_{k=1}^{r} \lambda_k \mathbf{U}(i_1, k) \mathbf{V}(k, i_2). \tag{A.7}$$

En conséquence, chaque élément d'un tenseur d'ordre supérieur $\mathcal{X}$ peut être représenté par

$$\mathcal{X}(i_1, i_2, \ldots, i_N) \overset{\text{TT}}{=} \sum_{k_1, k_2, \ldots, k_{N-1}}^{r_1, r_2, \ldots, r_{N-1}} \mathcal{G}_1(1, i_1, k_1) \mathcal{G}_2(k_1, i_2, k_2) \ldots \mathcal{G}_N(k_{N-1}, i_N, 1).$$
$$\tag{A.8}$$

où $\mathcal{G}_n$ est un $r_{n-1} \times I_n \times r_n$ tenseur avec $n = 1, 2, \ldots, N-1$ et $r_0 = r_N = 1$. Nous nous référons au modèle de représentation (A.8) en tant que train de tenseurs (TT). Comme CP, le format TT offre un modèle d'économie de mémoire pour représenter les tenseurs d'ordre élevé. Comme Tucker, la décomposition TT et le rang TT $\mathbf{r} = [r_1, r_2, \ldots, r_{N-1}]$ de tout tenseur $\mathcal{X}$ peuvent être calculés numériquement de manière stable et efficace.

Décomposition t-SVD: Enfin, une autre extension de SVD aux tenseurs

d'ordre élevé est le tenseur SVD (t-SVD) qui est de la forme suivante:

$$
\mathcal{X} \overset{\text{t-SVD}}{=} \underbrace{\mathcal{U}}_{\text{orthogonal}} * \underbrace{\mathcal{G}}_{f\text{-diagonal}} * \underbrace{\mathcal{V}}_{\text{orthogonal}} , \tag{A.9}
$$

où $\mathcal{U}$ et $\mathcal{V}$ sont des tenseurs unitaires, et $\mathcal{G}$ est un rectangle f-tenseur diagonal dont les tranches frontales sont des matrices diagonales [17]. Intuitivement, le modèle t-SVD (A.9) partage la même forme avec le SVD in (A.1). Cependant, en raison du t-produit " $*$ ", le cadre algébrique utilisé dans le t-SVD est assez différent de l'algèbre (multi)-linéaire classique dans d'autres types de TD et SVD. Par exemple, la plupart des ses calculs sont effectués dans le domaine de Fourier Sous le format t-SVD, le tubal-rank qui est égal au nombre de tubes non nuls de $\mathcal{G}$ est utilisé pour définir le LRA des tenseurs de la même manière que le SVD.

A.1.2 Approximation de Rang Inférieur en Ligne: Du Sous-espace au Suivi Tensoriel

Dans le cadre en ligne, des échantillons de données sont collectés en continu avec le temps. En conséquence, le recalcul des méthodes LRA par lots (par exemple, les algorithmes SVD ou TD par lots) à chaque pas de temps devient inefficace en raison de leur grande complexité et de la variation temporelle, c'est-à-dire de la dérive concept/distribution. Cela a conduit à définir une variante de la LRA appelée LRA en ligne (adaptative) dans laquelle nous pouvons vouloir suivre le processus sous-jacent qui génère des données en continu dans le temps.

Lorsque les observations arrivant à chaque instant sont unidimensionnelles (c'est-à-dire vectorielles), l'intérêt principal de la LRA en ligne est d'estimer le sous-espace principal qui couvre de manière compacte ces observations dans le temps. Plus précisément, on parle de problème de suivi de sous-espace (ST) dans le traitement du signal, qui a été développé pendant plus de trois décennies [19–21]. En général, à l'arrivée des nouvelles données $\mathbf{y}_t \in \mathbb{R}^{I_1 \times 1}$ au temps t, la matrice de sous-espace $\mathbf{U}_t \in \mathbb{R}^{I_1 \times r}$ peut être dérivé de l'analyse du spectre de la matrice de covariance suivante

$$
\mathbf{C}_t = \sum_{\tau = t - L_t + 1}^{t} \beta^{t-\tau} \mathbf{y}_\tau \mathbf{y}_\tau^\top, \tag{A.10}
$$

où L_t est la longueur de la fenêtre et $0 < \beta \leq 1$ est le facteur d'oubli [20]. Lorsque $L_t = t$ et $\beta = 1$, $\mathbf{C}_t$ dans (A.10) se résume à la matrice de covariance d'échantillon classique. Plus précisément, dans une connexion au batch LRA utilisant SVD, le vecteur $\mathbf{y}_t$ peut être vu comme la t-ième colonne de la matrice
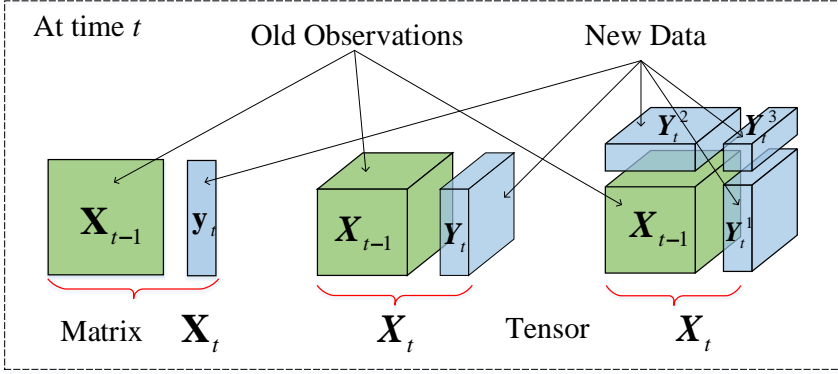
Figure A.3: Données en continu.

sous-jacente $\mathbf{X}_t = [\mathbf{X}_{t-1} \ \ \mathbf{y}_t]$, voir Fig. A.3 pour une illustration. La matrice de sous-espace $\mathbf{U}_t$ joue le rôle de matrice vectorielle singulière gauche de $\mathbf{X}_t$, tandis que le vecteur de coefficients $\mathbf{w}_t = \mathbf{U}_t^\top \mathbf{y}_t$ est bien la t-ième ligne de la matrice $\mathbf{V}\boldsymbol{\Lambda}$ dans l'expression SVD (A.1). Selon le choix de $\mathbf{C}_t$ et la technique d'estimation de sous-espace, nous pouvons obtenir plusieurs algorithmes de suivi de sous-espace.

Lorsque les observations arrivant à chaque instant sont multidimension-nelles (c'est-à-dire tensorielles), le LRA en ligne s'avère être un suivi tenseur qui peut être considéré comme une généralisation du suivi subspatial. En particulier, nous souhaitons estimer le dictionnaire de tenseurs (par exemple, le(s) tenseur(s) central(s) et les facteurs de tenseur) qui génère les données de flux sous-jacentes $\boldsymbol{\mathcal{X}}_t$ au fil du temps:

$$\boldsymbol{\mathcal{X}}_t = \begin{cases} \boldsymbol{\mathcal{X}}_{t-1} \boxplus \boldsymbol{\mathcal{Y}}_t & \text{si streaming mono-aspect} \\ \boldsymbol{\mathcal{X}}_{t-1} \cup \boldsymbol{\mathcal{Y}}_t & \text{si streaming multi-aspects} \end{cases}, \tag{A.11}$$

où " $\boxplus$ " et " $\cup$ " désignent la concaténation du tenseur et l'opérateur d'union, tandis que $\boldsymbol{\mathcal{X}}_{t-1}$ et $\boldsymbol{\mathcal{Y}}_t$ représentent respectivement l'ancienne et la nouvelle observations. Le modèle "single-aspect streaming" et le modèle "multi-aspect streaming" sont, respectivement, dédiés à représenter des flux de données ayant une dimension et des dimensions multiples variant avec le temps. Lorsque de nouveaux échantillons de données arrivent, le dictionnaire de tenseurs de $\boldsymbol{\mathcal{X}}_t$ doit être mis à jour de manière incrémentielle sans réutiliser les algorithmes TD par lots. Semblable au suivi de sous-espace, nous pouvons également obtenir de nombreux algorithmes de suivi de tenseurs basés sur différents formats de tenseurs, modèles de flux et techniques d'optimisation. Les lecteurs sont renvoyés au chapitre 5 pour un aperçu complet des algo-rithmes de suivi de tenseur de pointe.

Ces dernières années, l'explosion des flux de données volumineuses a posé des défis importants au problème de la LRA en ligne. Par exemple, l'efficacité et la robustesse sont très importantes lorsque nous traitons des données en continu dans des dimensions élevées. De nombreux résultats théoriques dans la théorie des matrices aléatoires (par exemple, [22–24]) ont indiqué que la matrice de covariance de l'échantillon (SCM) n'est pas un estimateur efficace de la matrice de covariance réelle dans l'échantillon de grande dimension et de faible taille régime où les ensembles de données sont massifs à la fois en dimension et en taille d'échantillon. Cependant, la plupart des méthodes de suivi de sous-espace de pointe dans la littérature sont principalement basées sur l'analyse spectrale du SCM, et donc, elles ne sont pas efficaces dans un tel régime. En parallèle, les valeurs aberrantes clairsemées et les données manquantes deviennent de plus en plus omniprésentes dans les applications de streaming modernes [6]. Les valeurs aberrantes éparses sont des points de données qui semblent être incohérents ou qui présentent un comportement anormal différent des autres. Des données manquantes sont souvent rencontrées lors de l'acquisition et de la collecte. Les valeurs aberrantes éparses et les données manquantes peuvent entraîner plusieurs problèmes pour la découverte des connaissances à partir des données en général et des flux de données en particulier, voir Fig. A.4 pour une illustration de l'impact des valeurs aberrantes sur l'analyse en composantes principales (ACP) standard qui utilise spécifiquement SVD dans son calcul. Par conséquent, cela nécessite des algorithmes robustes capables de gérer de telles corruptions de données dans le temps. De plus, des algorithmes de suivi évolutifs sont toujours souhaitables pour gérer les flux de données modernes, en particulier pour les flux de données à grande échelle et hautement multidimensionnels. Comme indiqué plus loin, la plupart des algorithmes de suivi existants sont d'une complexité élevée en ce qui concerne à la fois le calcul et le stockage en mémoire. En conséquence, il est essentiel de développer des techniques de suivi efficaces et évolutives à faible coût. Dans ce travail, nous visons à développer des algorithmes de suivi efficients et efficaces qui ont la capacité de faire face à de tels défis.

A.2 Description de la Thèse

A.2.1 Sommaire et Contributions de la Thèse

Le reste de ma thèse est organisé en deux grandes parties traitant respectivement du suivi des sous-espaces et du suivi des tenseurs, suivies de la conclusion et des perspectives, veuillez consulter Fig. 1.6 pour un aperçu.
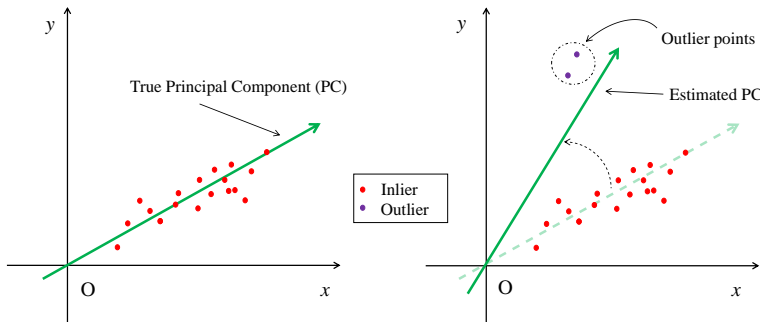
Figure A.4: Effet des valeurs aberrantes sur la norme PCA

Partie I: Suivi de Sous-espace

Dans le Chapitre 2, nous fournissons un bref aperçu des récents algorithmes robustes de suivi de sous-espace qui ont été principalement développés au cours de la dernière décennie. En particulier, nous commencons par introduire les idées de base du problème de suivi de sous-espace. Nous mettons ensuite en évidence les principales classes d'algorithmes pour traiter les bruits non gaussiens (par exemple, les valeurs aberrantes éparses, le bruit impulsif et le bruit coloré). Ces dernières années ont également vu la généralisation de l'analyse de données de grande dimension dans laquelle des méthodes basées sur la représentation clairsemée sont appliquées avec succès à de nombreuses applications de traitement du signal. En conséquence, les algorithmes de suivi de sous-espace clairsemé de l'état de l'art y sont également passés en revue.

Dans Chapitre 3, nous proposons un nouvel algorithme, à savoir PETRELS-ADMM, pour traiter le suivi de sous-espace en présence de valeurs aberrantes et de données manquantes. L'approche proposée consiste en deux étapes principales: le rejet des valeurs aberrantes et l'estimation du sous-espace. Dans la première étape, la méthode des multiplicateurs à direction alternée (ADMM) est efficacement exploitée pour détecter les valeurs aberrantes affectant les données observées. Dans la deuxième étape, nous proposons une version améliorée de l'algorithme d'estimation et de suivi parallèles par les moindres carrés récursifs (PETRELS) pour mettre à jour le sous-espace sous-jacent dans le contexte des données manquantes. Nous présentons ensuite une analyse de convergence théorique de PETRELS-ADMM qui montre qu'il génère une séquence de solutions de sous-espaces convergeant vers l'optimum de son homologue batch. L'efficacité de l'algorithme proposé, par rapport aux algorithmes de pointe, est illustrée à la fois sur des données simulées et réelles.

Dans Chapitre 4, nous développons une nouvelle méthode efficace prouvable appelée OPIT pour suivre le sous-espace principal clairsemé des flux

de données au fil du temps. En particulier, OPIT introduit une nouvelle variante adaptative d'itération de puissance avec un espace et une complexité de calcul linéaires à la dimension des données. De plus, un nouvel opérateur de seuillage basé sur les colonnes est développé pour régulariser la parcimonie du sous-espace. Utilisant à la fois les avantages de l'itération de puissance et de l'opération de seuillage, OPIT est capable de suivre le sous-espace sous-jacent à la fois en régime classique et en régime de grande dimension. Nous présentons également un résultat théorique sur sa convergence pour vérifier sa consistance en grandes dimensions. Plusieurs expériences sont menées sur des données synthétiques et réelles pour démontrer la capacité de suivi d'OPIT.

Partie II: Suivi Tensoriel

Dans le Chapitre 5, nous proposons une étude contemporaine et complète des différents types de techniques de suivi des tenseurs. Nous classons en particulier les méthodes de pointe en trois groupes principaux : les décompositions CP en continu, les décompositions en continu de Tucker et les décompositions en continu sous d'autres formats de tenseurs (c'est-à-dire, train de tenseurs, t-SVD et BTD). Dans chaque groupe, nous divisons en outre les algorithmes existants en sous-catégories en fonction de leur cadre d'optimisation principal et des architectures de modèles. Plus précisément, quatre groupes principaux d'algorithmes de décomposition CP en continu ont été mis en évidence, y compris la descente en coordonnées de bloc basée sur le sous-espace, l'inférence bayésienne et les décompositions en continu multi-aspects. Nous catégorisé la décomposition actuelle de Tucker en streaming méthodes en trois grandes classes en fonction de leur modèle architecture. Il s'agit de l'apprentissage en ligne du dictionnaire de tenseurs, du suivi du sous-espace tenseur et des décompositions en continu multi-aspects. Enfin, un bref aperçu des méthodes existantes capables de suivre les tenseurs sous les formats TT, BTD et t-SVD est présenté.

Dans Chapitre 6, nous proposons trois nouveaux algorithmes adaptatifs pour suivre les tenseurs de flux d'ordre supérieur avec le temps, y compris ACP, ATD et RACP. Sous le format CP, ACP minimise une fonction de coût des moindres carrés récursive pondérée exponentiellement pour obtenir les facteurs tensoriels de manière efficace, grâce au cadre de minimisation alternatif et à la technique d'esquisse aléatoire. Sous le format Tucker, ATD suit d'abord les sous-espaces sous-jacents de faible dimension couvrant les facteurs tensoriels, puis estime le tenseur central à l'aide d'une approximation stochastique a. Les deux algorithmes ACP et ATD sont rapides et parfaitement capables de suivre les tenseurs de flux à partir d'observations incomplètes. Lorsque les observations sont corrompues par des valeurs aberrantes éparses,

nous introduisons l'algorithme dit RACP robuste aux corruptions grossières. En particulier, RACP effectue d'abord le rejet des valeurs aberrantes en ligne pour détecter et supprimer avec précision les valeurs aberrantes clairsemées, puis effectue un suivi des facteurs tensoriels pour mettre à jour efficacement les facteurs tensoriels. L'analyse de convergence pour trois algorithmes est établie dans le sens où la séquence de solutions générées converge asymptotiquement vers un point stationnaire de la fonction objectif. Des expériences approfondies sont menées sur des données synthétiques et réelles pour démontrer l'efficacité des algorithmes proposés par rapport aux algorithmes adaptatifs de pointe.

Dans le Chapitre 7, nous introduisons trois nouvelles méthodes pour le problème de la décomposition en continu des trains de tenseurs. La première méthode appelée TT-FOA est capable de suivre avec une grande précision les composantes de rang inférieur des tenseurs d'ordre élevé à partir de données bruitées et de grande dimension, même lorsqu'elles proviennent d'observations dépendant du temps. La deuxième méthode appelée ATT est particulièrement conçue pour gérer les tenseurs de flux incomplets. ATT est évolutif, efficace et apte à estimer les composants de faible rang TT des tenseurs de flux. En outre, ATT peut prendre en charge l'informatique parallèle et distribuée. Pour traiter les valeurs aberrantes éparses, nous proposons le soi-disant ROBOT qui signifie ROBust Online Tensor-Train decomposition. Techniquement, ROBOT a la capacité de suivre les tenseurs de streaming à partir de flux imparfaits (c'est-à-dire en raison du bruit, des valeurs aberrantes et des données manquantes) ainsi que de suivre leur variation temporelle dans des environnements dynamiques.

Conclusion et Perspectives

Le chapitre 8 conclut la thèse avec nos principaux résultats et un aperçu des travaux futurs. En particulier, nous présentons plusieurs défis de recherche et problèmes ouverts qui devraient être pris en compte pour le développement du suivi de la composante de rang inférieur des flux de données à l'avenir. Il s'agit de l'imperfection et de la corruption des données; classement et suivi; suivi de tenseur efficace et évolutif; et d'autres aspects tels que l'analyse théorique, les données symboliques et le suivi sous des formats de tenseur moins courants. Des solutions possibles à ces défis sont également discutées.

A.2.2 Liste des Publications

La plupart des résultats ci-dessus ont été publiés/soumis dans les articles suivants:

Articles de Journaux:

[25] L. T. Thanh, N. V. Dung, N. L. Trung and K. Abed-Meraim, *"Robust Subspace Tracking With Missing Data and Outliers: Novel Algorithm With Convergence Guarantee"*, **IEEE Trans. Signal Process.**, vol. 69, pp. 2070–2085, 2021.

[26] L. T. Thanh, N. V. Dung, N. L. Trung and K. Abed-Meraim, *"Robust Subspace Tracking Algorithms in Signal Processing: A Brief Survey"*, **REV J. Elect. Commun.**, vol. 11, no. 1–2, pp. 15–25, 2021.

[27] L. T. Thanh, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"Robust Tensor Tracking with Missing Data and Outliers: Novel Adaptive CP Decomposition and Convergence Analysis"*, **IEEE Trans. Signal Process.**, vol. 70, pp. 4305–4320, 2022.

[28] L. T. Thanh, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"A Contemporary and Comprehensive Survey on Streaming Tensor Decomposition"*, **IEEE Trans. Knowl. Data. Eng.**, 2022 (in press).

[29] L. T. Thanh, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"OPIT: A Simple and Effective Method for Sparse Subspace Tracking in High-dimension and Low-sample-size Context"*, **IEEE Trans. Signal Process.**, 2022 (submitted).

[30] L. T. Thanh, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"Tracking Online Low-Rank Approximations of Higher-Order Incomplete Streaming Tensors"*, **Elsevier Patterns**, 2022 (submitted).

[31] L. T. Thanh, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"Streaming Tensor-Train Decomposition With Missing Data"*, **Elsevier Signal Process.**, 2022 (submitted).

Conférence:

[32] L. T. Thanh, K. Abed-Meraim, N. L. Trung and R. Boyer, *"Adaptive Algorithms for Tracking Tensor-Train Decomposition of Streaming Tensors"*, in **Proc. 28th EUSIPCO**, 2020, pp. 995–999.

[33] L. T. Thanh, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"A Fast Randomized Adaptive CP Decomposition for Streaming Tensors"*, in **Proc. 46th ICASSP**, 2021, pp. 2910–2914.

[34] L. T. Thanh, K. Abed-Meraim, A. Hafiane and N. L. Trung, *"Sparse Subspace Tracking in High Dimensions"*, in **Proc. 47th ICASSP**, 2022, pp. 5892–5896.

[35] L. T. Thanh, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"Robust Tensor Tracking With Missing Data Under Tensor-Train Format"*, in **Proc. 30th EUSIPCO**, 2022, pp. 832–836.

[36] L. T. Thanh, T. T. Duy, K. Abed-Meraim, N. L. Trung and A. Hafiane, *"Robust Online Tucker Dictionary Learning from Multidimensional Data Streams"*, in **Proc. 14th APSIPA-ASC**, 2022, pp. 1815–1820.

Contributions en Dehors du Champ de la Thèse

Au cours de mes études de doctorat, j'ai également apporté d'autres contributions à l'identification des systèmes aveugles qui ne sont pas incluses dans cette thèse:

[37] L. T. Thanh, K. Abed-Meraim and N. L. Trung, *"Misspecified Cramer-Rao Bounds for Blind Channel Estimation under Channel Order Misspecification"*, **IEEE Trans. Signal Process.**, vol. 69, pp. 5372–5385, 2021.

[38] L. T. Thanh, K. Abed-Meraim and N. L. Trung, *"Performance Lower Bounds of Blind System Identification Techniques in the Presence of Channel Order Estimation Error"*, in **Proc. 29th EUSIPCO**, 2021, pp. 1646–1650.

[39] O. Rekik, A. Mokraoui, T. T. T Quynh, L. T. Thanh and K. Abed-Meraim. *"Side Information Effect on Semi-Blind Channel Identification for MIMO-OFDM Communications Systems"*, in **Proc. 55th ASILOMAR 2021**, pp. 443–448.

Bibliography

[1] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," *Inf. Sci.*, vol. 275, pp. 314–347, 2014.

[2] T. Kolajo, O. Daramola, and A. Adebiyi, "Big data stream analysis: A systematic literature review," *J. Big Data*, vol. 6, no. 1, pp. 1–30, 2019.

[3] M. Bahri, A. Bifet, J. Gama, H. M. Gomes, and S. Maniu, "Data stream analysis: Foundations, major tasks and tools," *WIREs Data Min. Knowl. Discov.*, vol. 11, no. 3, p. 1405, 2021.

[4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surv. Tutor.*, vol. 17, no. 4, pp. 2347–2376, 2015.

[5] A. A. Safaei, "Real-time processing of streaming big data," *Real-Time Syst.*, vol. 53, no. 1, pp. 1–44, 2017.

[6] T. Akidau, S. Chernyak, and R. Lax, *Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing*, 2018.

[7] I. Markovsky, *Low-Rank Approximation: Algorithms, Implementation, Applications*, 2019.

[8] G. W. Stewart, "On the early history of the singular value decomposition," *SIAM Rev.*, vol. 35, no. 4, pp. 551–566, 1993.

[9] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2012.

[10] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.

[11] N. D. Sidiropoulos, L. D. 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.

[12] A. Cichocki, N. Lee, I. V. Oseledets, A.-H. Phan, Q. Zhao, and D. P. Mandic, "Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions," *Found. Trends Mach. Learn.*, vol. 9, no. 4-5, pp. 249–429, 2016.

[13] Y. Liu, J. Liu, Z. Long, and C. Zhu, *Tensor Computation for Data Analysis*, 2022.

[14] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an explanatory multimodal factor analysis," *UCLA Work. Pap. Phon.*, vol. 16, no. 1-84, 1970.

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

[16] I. V. Oseledets, "Tensor-train decomposition," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2295–2317, 2011.

[17] M. E. Kilmer and C. D. Martin, "Factorization strategies for third-order tensors," *Linear Algebra Appl.*, vol. 435, no. 3, pp. 641–658, 2011.

[18] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms—Part II: Definitions and uniqueness," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1033–1066, 2008.

[19] P. Comon and G. H. Golub, "Tracking a few extreme singular values and vectors in signal processing," *Proc. IEEE*, vol. 78, no. 8, pp. 1327–1343, 1990.

[20] J.-P. Delmas, "Subspace tracking for signal processing," in *Adaptive Signal Processing: Next Generation Solutions*, 2010, pp. 211–270.

[21] N. Vaswani, T. Bouwmans, S. Javed, and P. Narayanamurthy, "Robust subspace learning: Robust PCA, robust subspace tracking, and robust subspace recovery," *IEEE Signal Process. Mag.*, vol. 35, no. 4, pp. 32–55, 2018.

[22] N. El. Karoui, "Spectrum estimation for large dimensional covariance matrices using random matrix theory," *Ann. Stat.*, vol. 36, no. 6, pp. 2757–2790, 2008.

[23] X. Mestre, "On the asymptotic behavior of the sample estimates of eigenvalues and eigenvectors of covariance matrices," *IEEE Trans. Signal Process.*, vol. 56, no. 11, pp. 5353–5368, 2008.

[24] R. Vershynin, "How close is the sample covariance matrix to the actual covariance matrix?" *J. Theor. Probab.*, vol. 25, no. 3, pp. 655–686, 2012.

[25] L. T. Thanh, N. V. Dung, N. L. Trung, and K. Abed-Meraim, "Robust subspace tracking with missing data and outliers: Novel algorithm with convergence guarantee," *IEEE Trans. Signal Process.*, vol. 69, pp. 2070–2085, 2021.

[26] ——, "Robust subspace tracking algorithms in signal processing: A brief survey," *REV J. Elect. Commun.*, vol. 11, no. 1-2, pp. 16–25, 2021.

[27] L. T. Thanh, K. Abed-Meraim, N. Linh Trung, and A. Hafiane, "Robust tensor tracking with missing data and outliers: Novel adaptive CP decomposition and convergence analysis," vol. 70, pp. 4305–4320, 2022.

[28] ——, "A Contemporary and Comprehensive Survey on Streaming Tensor Decomposition," *IEEE Trans. Knowl. Data Eng.*, 2022 (submitted).

[29] L. T. Thanh, K. Abed Meraim, N. L. Trung, and A. Hafiane, "OPIT: A Simple and Effective Method for Sparse Subspace Tracking in High-dimension and Low-sample-size Context," *IEEE Trans. Signal Process.*, 2022 (submitted).

[30] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "Tracking dynamic low-rank approximations of incomplete high-order streaming tensors," *Elsevier Patterns*, 2022 (submitted).

[31] L. T. Thanh, K. Abed Meraim, N. Linh Trung, and A. Hafiane, "Streaming tensor-train decomposition with missing data," *Signal Process.*, 2022 (submitted).

[32] L. T. Thanh, K. Abed-Meraim, N. Linh-Trung, and R. Boyer, "Adaptive algorithms for tracking tensor-train decomposition of streaming tensors," in *Eur. Signal Process. Conf.*, 2020, pp. 995–999.

[33] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "A fast randomized adaptive CP decomposition for streaming tensors," in *IEEE Int. Conf. Acoust. Speech Signal Process.*, 2021, pp. 2910–2914.

[34] L. T. Thanh, K. Abed-Meraim, A. Hafiane, and N. L. Trung, "Sparse subspace tracking in high dimensions," in *IEEE Int. Conf. Acoust. Speech Signal Process.*, 2022, pp. 5892–5896.

[35] L. T. Thanh, K. Abed-Meraim, N. Linh Trung, and A. Hafiane, "Robust tensor tracking with missing data under tensor-train format," in *Eur. Signal. Process. Conf.*, 2022, pp. 832–836.

[36] L. T. Thanh, K. Abed-Meraim, N. L. Trung, and A. Hafiane, "Robust online tucker dictionary learning from multidimensional data streams,"

in *Proc. 14th Asia-Pacific Signal Inf. Process. Assoc. Ann. Summit Conf.*, 2022.

[37] L. T. Thanh, K. Abed-Meraim, and N. L. Trung, "Misspecified Cramer–Rao Bounds for Blind Channel Estimation Under Channel Order Misspecification," *IEEE Trans. Signal Process.*, vol. 69, pp. 5372–5385, 2021.

[38] ——, "Performance lower bounds of blind system identification techniques in the presence of channel order estimation error," in *Eur. Signal Process. Conf.*, 2021, pp. 1646–1650.

[39] O. Rekik, A. Mokraoui, T. T. Thuy Quynh, T.-T. Le, and K. Abed-Meraim, "Side Information Effect on Semi-Blind Channel Identification for MIMO-OFDM Communications Systems," in *2021 55th Asilomar Conference on Signals, Systems, and Computers*, 2021, pp. 443–448.

[40] I. Jolliffe, *Principal Component Analysis*, 2002.

[41] L. Balzano, Y. Chi, and Y. M. Lu, "Streaming PCA and Subspace Tracking: The Missing Data Case," *Proceed. IEEE*, vol. 106, no. 8, pp. 1293–1310, 2018.

[42] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *J. ACM*, vol. 58, no. 3, p. 11, 2011.

[43] N. Vaswani, Y. Chi, and T. Bouwmans, "Rethinking PCA for modern data sets: Theory, algorithms, and applications," *Proc. IEEE*, vol. 106, no. 8, pp. 1274–1276, 2018.

[44] I. T. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," *Philos. Trans. Royal Soc. A*, vol. 374, no. 2065, p. 20150202, 2016.

[45] C. Wang, Y. C. Eldar, and Y. M. Lu, "Subspace estimation from incomplete observations: A high-dimensional analysis," *IEEE J. Sel. Top. Signal Process.*, vol. 12, no. 6, pp. 1240–1252, 2018.

[46] N. Vaswani and P. Narayanamurthy, "Static and dynamic robust PCA and matrix completion: A review," *Proc. IEEE*, vol. 106, no. 8, pp. 1359–1379, 2018.

[47] G. Lerman and T. Maunu, "An overview of robust subspace recovery," *Proc. IEEE*, vol. 106, no. 8, pp. 1380–1410, 2018.

[48] S. X. Wu, H. Wai, L. Li, and A. Scaglione, "A review of distributed algorithms for principal component analysis," *Proc. IEEE*, vol. 106, no. 8, pp. 1321–1340, 2018.

[49] H. Zou and L. Xue, "A selective overview of sparse principal component analysis," *Proc. IEEE*, vol. 106, no. 8, pp. 1311–1320, 2018.

[50] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the Grassmannian for online foreground and background separation in subsampled video," in *IEEE Conf. Comput. Vis. Pattern Recogn.* IEEE, 2012, pp. 1568–1575.

[51] J. Xu, V. K. Ithapu, L. Mukherjee, J. M. Rehg, and V. Singh, "GOSUS: Grassmannian online subspace updates with structured-sparsity," in *IEEE Int. Conf. Comput. Vis.*, 2013, pp. 3376–3383.

[52] F. Seidel, C. Hage, and M. Kleinsteuber, "pROST: A smoothed lp-Norm robust online subspace tracking method for background subtraction in video," *Mach. Vis. Appl.*, vol. 25, no. 5, pp. 1227–1240, 2014.

[53] C. Hage and M. Kleinsteuber, "Robust PCA and subspace tracking from incomplete observations using ℓ_0-Surrogates," *Comput. Stat.*, vol. 29, no. 3-4, pp. 467–487, 2014.

[54] J. Shen, H. Xu, and P. Li, "Online optimization for max-norm regularization," in *Adv. Neural Inf. Process. Syst.*, 2014, pp. 1718–1726.

[55] H. Mansour and X. Jiang, "A robust online subspace estimation and tracking algorithm," in *IEEE Int. Conf. Acoust. Speech Signal Process.*, 2015, pp. 4065–4069.

[56] S. Chouvardas, Y. Kopsinis, and S. Theodoridis, "An Adaptive Projected Subgradient based algorithm for robust subspace tracking," in *IEEE Int. Conf. Acoust. Speech Signal Process.*, 2014, pp. 5497–5501.

[57] ——, "Robust subspace tracking with missing entries: The set-theoretic approach," *IEEE Trans. Signal Process.*, vol. 63, no. 19, pp. 5060–5070, 2015.

[58] J. Zhan, B. Lois, H. Guo, and N. Vaswani, "Online (and offline) robust PCA: Novel algorithms and performance guarantees," in *Artif. Intell. Stat.*, 2016, pp. 1488–1496.

[59] B. Hong, L. Wei, Y. Hu, D. Cai, and X. He, "Online robust principal component analysis via truncated nuclear norm regularization," *Neurocomput.*, vol. 175, pp. 216–222, 2016.

[60] K. G. Quach, C. N. Duong, K. Luu, and T. D. Bui, "Non-convex online robust PCA: Enhance sparsity via $_p$-Norm minimization," *Comput. Vis. Image Underst.*, vol. 158, pp. 126–140, 2017.

[61] P. P. Markopoulos, M. Dhanaraj, and A. Savakis, "Adaptive L1-norm principal-component analysis with online outlier rejection," *IEEE J. Sel. Top. Signal Process.*, vol. 12, no. 6, pp. 1131–1143, 2018.

[62] N. Linh-Trung, V. D. Nguyen, M. Thameri, T. Minh-Chinh, and K. Abed-Meraim, "Low-complexity adaptive algorithms for robust subspace tracking," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 6, pp. 1197–1212, 2018.

[63] P. Narayanamurthy and N. Vaswani, "Provable dynamic robust PCA or robust subspace tracking," *IEEE Trans. Inf. Theory*, vol. 65, no. 3, pp. 1547–1577, 2019.

[64] P. Narayanamurthy, V. Daneshpajooh, and N. Vaswani, "Provable subspace tracking from missing data and matrix completion," *IEEE Trans. Signal Process.*, pp. 4245–4260, 2019.

[65] Y. Liu, K. Tountas, D. A. Pados, S. N. Batalama, and M. J. Medley, "L1-subspace tracking for streaming data," *Pattern Recog.*, vol. 97, p. 106992, 2020.

[66] X. Jia, X. Feng, W. Wang, H. Huang, and C. Xu, "Online Schatten quasi-norm minimization for robust principal component analysis," *Inf. Sci.*, vol. 476, pp. 83–94, 2019.

[67] P. Narayanamurthy and N. Vaswani, "Fast robust subspace tracking via PCA in sparse data-dependent noise," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 3, pp. 723–744, 2020.

[68] R. Chakraborty, S. Hauberg, and B. C. Vemuri, "Intrinsic Grassmann averages for online linear and robust subspace learning," in *IEEE Conf. Comput. Vis. Pattern Recogn.*, 2017, pp. 6196–6204.

[69] R. Chakraborty, L. Yang, S. Hauberg, and B. Vemuri, "Intrinsic Grassmann averages for online linear, robust and nonlinear subspace learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–1, 2020.

[70] L. T. Thanh, N. V. Dung, N. L. Trung, and K. Abed Meraim, "Robust subspace tracking with missing data and outliers via ADMM," in *European Signal Process. Conf.*, 2019, pp. 1–5.

[71] S. Hauberg, A. Feragen, R. Enficiaud, and M. J. Black, "Scalable robust principal component analysis using Grassmann averages," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 11, pp. 2298–2311, 2016.

[72] L. Balzano, R. Nowak, and B. Recht, "Online identification and tracking of subspaces from highly incomplete information," in *Allerton Conf. Commun. Control Comput.*, 2010, pp. 704–711.

[73] Y. Chi, Y. C. Eldar, and R. Calderbank, "PETRELS: Parallel subspace estimation and tracking by recursive least squares from partial observations," *IEEE Trans. Signal Process.*, vol. 61, no. 23, pp. 5947–5959, 2013.

[74] B. Yang, "Projection approximation subspace tracking," *IEEE Trans. Signal Process.*, vol. 43, no. 1, pp. 95–107, 1995.

[75] K. L. Blackard, T. S. Rappaport, and C. W. Bostian, "Measurements and models of radio frequency impulsive noise for indoor wireless communications," *IEEE J. Sel. Areas Commun.*, vol. 11, no. 7, pp. 991–1001, 1993.

[76] W. Ebel and W. Tranter, "The performance of Reed-Solomon codes on a bursty-noise channel," *IEEE Trans. Commun.*, vol. 43, no. 2/3/4, pp. 298–306, 1995.

[77] Kung Yao, "A representation theorem and its applications to spherically-invariant random processes," *IEEE Trans. Inf. Theory*, vol. 19, no. 5, pp. 600–608, 1973.

[78] E. Ollila, D. E. Tyler, V. Koivunen, and H. V. Poor, "Complex elliptically symmetric distributions: Survey, new results and applications," *IEEE Trans. Signal Process.*, vol. 60, no. 11, pp. 5597–5625, 2012.

[79] C. L. Nikias and M. Shao, *Signal Processing with Alpha-Stable Distributions and Applications*, 1995.

[80] P. G. Georgiou, P. Tsakalides, and C. Kyriakakis, "Alpha-stable modeling of noise and robust time-delay estimation in the presence of impulsive noise," *IEEE Trans. Multimedia*, vol. 1, no. 3, pp. 291–301, 1999.

[81] S.-C. Chan, Y. Wen, and K.-L. Ho, "A robust past algorithm for subspace tracking in impulsive noise," *IEEE Trans. Signal Process.*, vol. 54, no. 1, pp. 105–116, 2006.

[82] J. Zhang and T.-s. Qiu, "A robust correntropy based subspace tracking algorithm in impulsive noise environments," *Digit. Signal Process.*, vol. 62, pp. 168–175, 2017.

[83] S. Luan, T. Qiu, L. Yu, J. Zhang, A. Song, and Y. Zhu, "BNC-based projection approximation subspace tracking under impulsive noise," *IET Radar Sonar Navig.*, vol. 11, no. 7, pp. 1055–1061, 2017.

[84] B. Liao, Z. Zhang, and S.-C. Chan, "A new robust Kalman filter-based subspace tracking algorithm in an impulsive noise environment," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 57, no. 9, pp. 740–744, 2010.

[85] J.-f. ZHANG, T.-s. QIU, and S. LI, "A robust PAST algorithm based on maximum correntropy criterion for impulsive noise environments," *Acta Electonica Sin.*, vol. 43, no. 3, p. 483, 2015.

[86] J. Zhang and T. Qiu, "A novel tracking method for fast varying subspaces in impulsive noise environments," in *2016 Int. Conf. Signal Process. Commun. Syst.*, 2016, pp. 1–7.

[87] S. Chan, Z. Zhang, and Y. Zhou, "A new adaptive Kalman filter-based subspace tracking algorithm and its application to DOA estimation," in *IEEE Int. Symp. Circuits Systt.*, 2006, pp. 4 pp.–132.

[88] T. Gustafsson, "Instrumental variable subspace tracking using projection approximation," *IEEE Trans. Signal Process.*, vol. 46, no. 3, pp. 669–681, 1998.

[89] G. Mercère, L. Bako, and S. Lecœuche, "Propagator-based methods for recursive subspace model identification," *Signal Process.*, vol. 88, no. 3, pp. 468–491, 2008.

[90] S. Chan, H. Tan, and J. Lin, "A new variable forgetting factor and variable regularized square root extended instrumental variable PAST algorithm with applications," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 3, pp. 1886–1902, 2020.

[91] M. Chen and Z. Wang, "Subspace tracking in colored noise based on oblique projection," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 3, 2006, pp. III–III.

[92] F. Yger, M. Berar, G. Gasso, and A. Rakotomamonjy, "Oblique principal subspace tracking on manifold," in *IEEE Int. Conf. Acoust. Speech Signal Process.*, 2012, pp. 2429–2432.

[93] C. Wang and Y. M. Lu, "Online learning for sparse PCA in high dimensions: Exact dynamics and phase transitions," in *IEEE Inf. Theory Works.*, 2016, pp. 186–190.

[94] W. Yang and H. Xu, "Streaming Sparse Principal Component Analysis," in *Int. Conf. Mach. Learn.*, 2015, pp. 494–503.

[95] X. Yang, Y. Sun, T. Zeng, T. Long, and T. K. Sarkar, "Fast STAP method based on PAST with sparse constraint for airborne phased array radar," *IEEE Trans. Signal Process.*, vol. 64, no. 17, pp. 4550–4561, 2016.

[96] P. V. Giampouras, A. A. Rontogiannis, K. E. Themelis, and K. D. Koutroumbas, "Online sparse and low-rank subspace learning from incomplete data: A Bayesian view," *Signal Process.*, vol. 137, pp. 199–212, 2017.

[97] N. Lassami, K. Abed-Meraim, and A. Aïssa-El-Bey, "Low cost subspace tracking algorithms for sparse systems," in *Eur. Signal Process. Conf.*, 2017, pp. 1400–1404.

[98] N. Lassami, A. Aïssa-El-Bey, and K. Abed-Meraim, "Low cost sparse subspace tracking algorithms," *Signal Process.*, vol. 173, p. 107522, 2020.

[99] R. Badeau, G. Richard, and B. David, "Fast and stable YAST algorithm for principal and minor subspace tracking," *IEEE Trans. Signal Process.*, vol. 56, no. 8, pp. 3437–3446, 2008.

[100] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang, "A survey of sparse representation: Algorithms and applications," *IEEE Access*, vol. 3, pp. 490–530, 2015.

[101] T. T. Cai, Z. Ma, Y. Wu *et al.*, "Sparse PCA: Optimal rates and adaptive estimation," *Ann. Stat.*, vol. 41, no. 6, pp. 3074–3110, 2013.

[102] D. Papailiopoulos, A. Dimakis, and S. Korokythakis, "Sparse PCA through low-rank approximations," in *Int. Conf. Mach. Learn.*, 2013, pp. 747–755.

[103] V. Q. Vu, J. Cho, J. Lei, and K. Rohe, "Fantope projection and selection: A near-Optimal convex relaxation of sparse PCA," in *Adv. Neural Inf. Process. Syst.*, 2013, pp. 2670–2678.

[104] N. Lassami, A. Aïssa-El-Bey, and K. Abed-Meraim, "Fast sparse subspace tracking algorithm based on shear and givens rotations," in *Asilomar Conf. Signals Syst. Comput.*, 2019, pp. 1667–1671.

[105] A. Tulay and H. Simon, *Adaptive Signal Processing: Next Generation Solutions*, 2010.

[106] M. Mardani, G. Mateos, and G. B. Giannakis, "Subspace learning and imputation for streaming big data matrices and tensors," *IEEE Trans. Signal Process.*, vol. 63, no. 10, pp. 2663–2677, 2015.

[107] N. V. Dung, K. Abed-Meraim, N. L. Trung, and R. Weber, "Generalized minimum noise subspace for array processing," *IEEE Trans. Signal Process.*, vol. 65, no. 14, pp. 3789–3802, 2017.

[108] S. Haghighatshoar and G. Caire, "Low-complexity massive MIMO subspace estimation and tracking from low-dimensional projections," *IEEE Trans. Signal Process.*, vol. 66, no. 7, pp. 1832–1844, 2018.

[109] S. Buzzi and C. D'Andrea, "Subspace tracking and least squares approaches to channel estimation in millimeter wave multiuser MIMO," *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 6766–6780, 2019.

[110] D. Zhang and L. Balzano, "Global convergence of a Grassmannian gradient descent algorithm for subspace estimation." in *Int. Conf. Artif. Intell. Stat.*, Cadiz, Spain, 2016, pp. 1460–1468.

[111] A. Gonen, D. Rosenbaum, Y. C. Eldar, and S. Shalev-Shwartz, "Subspace learning with partial information," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1821–1841, 2016.

[112] M. Shor and N. Levanon, "Performances of order statistics CFAR," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 27, no. 2, pp. 214–224, 1991.

[113] J. A. Tropp, "Just relax: Convex programming methods for identifying sparse signals in noise," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 1030–1051, 2006.

[114] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[115] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, "Optimal parameter selection for the alternating direction method of multipliers (ADMM): Quadratic problems," *IEEE Trans. Automat. Contr.*, vol. 60, no. 3, pp. 644–658, 2015.

[116] Y. Xu, M. Liu, Q. Lin, and T. Yang, "ADMM without a fixed penalty parameter: Faster convergence with new adaptive penalization," in *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1267–1277.

[117] W. Tian and X. Yuan, "An alternating direction method of multipliers with a worst-case $O(1/n^2)$ convergence rate," *Math. Comput.*, vol. 88, no. 318, pp. 1685–1713, 2019.

[118] N. Parikh and S. Boyd, "Proximal Algorithms," *Found. Trends Opt.*, vol. 1, no. 3, pp. 127–239, 2014.

[119] W. W. Hager, "Updating the inverse of a matrix," *SIAM Rev.*, vol. 31, no. 2, pp. 221–239, 1989.

[120] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, no. Jan, pp. 19–60, 2010.

[121] J. Feng, H. Xu, and S. Yan, "Online robust PCA via stochastic optimization," in *Adv. Neural Inf. Process. Syst.*, 2013, pp. 404–412.

[122] J. Shen, H. Xu, and P. Li, "Online optimization for max-norm regularization," *Mach. Learn.*, vol. 106, no. 3, pp. 419–457, 2017.

[123] G. Li and T. K. Pong, "Global convergence of splitting methods for nonconvex composite optimization," *SIAM J. Optim.*, vol. 25, no. 4, pp. 2434–2460, 2015.

[124] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *J. Sci. Comput.*, vol. 78, no. 1, pp. 29–63, 2019.

[125] L. Bottou, "Online learning and stochastic approximations," *-Line Learn. Neural Netw.*, vol. 17, no. 9, p. 142.

[126] A. W. Van der Vaart, *Asymptotic Statistics*, 2000.

[127] D. M. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," *J. Mach. Learn. Tech.*, vol. 2, no. 1, pp. 37–63, 2011.

[128] B. Vandereycken, "Low-rank matrix completion by Riemannian optimization," *SIAM J. Optim.*, vol. 23, no. 2, pp. 1214–1236, 2013.

[129] X. Yi, D. Park, Y. Chen, and C. Caramanis, "Fast algorithms for robust PCA via gradient descent," in *Adv. Neural Inf. Process. Syst.*, 2016, pp. 4152–4160.

[130] L. T. Nguyen, J. Kim, and B. Shim, "Low-rank matrix completion: A contemporary survey," *IEEE Access*, vol. 7, pp. 94 215–94 237, 2019.

[131] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "Changedetection.Net: A new change detection benchmark dataset," in *IEEE Conf. Comput. Vis. Pattern Recogn.*, 2012, pp. 1–8.

[132] S. Boyd and L. Vandenberghe, *Convex Optimization*, 2004.

[133] K. Knopp, *Theory and Application of Infinite Series*, 2013.

[134] S. Shalev-Shwartz and Y. Singer, "Online learning: Theory, algorithms, and applications," 2007.

[135] K. Fountoulakis and J. Gondzio, "A second-order method for strongly convex $_1$ -regularization problems," *Math. Program.*, vol. 156, no. 1-2, pp. 189–219, 2016.

[136] D. P. Bertsekas, "Nonlinear programming," *J. Oper. Res. Soc.*, vol. 48, no. 3, pp. 334–334, 1997.

[137] J. Gama, I. Zliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–37, 2014.

[138] N. El Karoui, "Operator norm consistent estimation of large-dimensional sparse covariance matrices," *Ann. Stat.*, vol. 36, no. 6, pp. 2717–2756, 2008.

[139] C. Lam and J. Fan, "Sparsistency and rates of convergence in large covariance matrix estimation," *Ann. Stat.*, vol. 37, no. 6B, p. 4254, 2009.

[140] I. M. Johnstone and A. U. Lu, "On consistency and sparsity for principal components analysis in high dimensions," *J. Am. Stat. Assoc.*, vol. 104, no. 486, pp. 682–693, 2009.

[141] P. J. Bickel and E. Levina, "Covariance regularization by thresholding," *Ann. Stat.*, vol. 36, no. 6, pp. 2577–2604, 2008.

[142] D. Shen, H. Shen, and J. S. Marron, "Consistency of sparse PCA in high dimension, low sample size contexts," *J. Mult. Anal.*, vol. 115, pp. 317–333, 2013.

[143] A. A. Arash and J. W. Martin, "High-dimensional analysis of semidefinite relaxations for sparse principal components," *Ann. Stat.*, vol. 37, no. 5B, pp. 2877–2921, 2009.

[144] M. Journée, Y. Nesterov, P. Richtárik, and R. Sepulchre, "Generalized power method for sparse principal component analysis." *J. Mach. Learn. Res.*, vol. 11, no. 2, pp. 517–553, 2010.

[145] Z. Ma, "Sparse principal component analysis and iterative thresholding," *Ann. Stat.*, vol. 41, no. 2, pp. 772–801, 2013.

[146] T. Cai, Z. Ren, and H. H. Zhou, "Estimating structured high-dimensional covariance and precision matrices: Optimal rates and adaptive estimation," *Electron. J. Stat.*, vol. 10, no. 1, pp. 1–59, 2016.

[147] P. Xiao and L. Balzano, "Online sparse and orthogonal subspace estimation from partial information," in *Allerton Conf. Commun. Control Comput.*, 2016, pp. 284–291.

[148] K. Abed-Meraim, S. Attallah, A. Chkeif, and Y. Hua, "Orthogonal Oja algorithm," *IEEE Signal Process. Lett.*, vol. 7, no. 5, pp. 116–119, 2000.

[149] Z. Allen-Zhu and Y. Li, "First efficient convergence for streaming k-PCA: A global, gap-free, and near-optimal rate," in *IEEE Ann. Symp. Found. Comput. Sci.*, 2017, pp. 487–492.

[150] Y. Hua, Y. Xiang, T. Chen, K. Abed-Meraim, and Y. Miao, "A new look at the power method for fast subspace tracking," *Digit. Signal Process.*, vol. 9, no. 4, pp. 297–314, 1999.

[151] K. Abed-Meraim, A. Chkeif, Y. Hua, and S. Attallah, "On a class of orthonormal algorithms for principal and minor subspace tracking," *J. VLSI Signal Process. Syst.*, vol. 31, no. 1, pp. 57–70, 2002.

[152] X. G. Doukopoulos and G. V. Moustakides, "Fast and stable subspace tracking," *IEEE Trans. Signal Process.*, vol. 56, no. 4, pp. 1452–1465, 2008.

[153] R. Wang, M. Yao, D. Zhang, and H. Zou, "A novel orthonormalization matrix based fast and stable DPM algorithm for principal and minor subspace tracking," *IEEE Trans. Signal Process.*, vol. 60, no. 1, pp. 466–472, 2011.

[154] R. Badeau, B. David, and G. Richard, "Fast approximated power iteration subspace tracking," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 2931–2941, 2005.

[155] Q. Wu, J. Zheng, Z. Dong, E. Panayirci, Z. Wu, and R. Qingnuobu, "An improved adaptive subspace tracking algorithm based on approximated power iteration," *IEEE Access*, vol. 6, pp. 43 136–43 145, 2018.

[156] N. Vaswani and P. Narayanamurthy, "Static and dynamic robust PCA and matrix completion: A review," *Proc. IEEE*, vol. 106, no. 8, pp. 1359–1379, 2018.

[157] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the Grassmannian for online foreground and background separation in subsampled video," in *IEEE Conf. Comput. Vis. Pattern Recogn.*, 2012, pp. 1568–1575.

[158] S.-C. Chan, Y. Wen, and K.-L. Ho, "A robust PAST algorithm for subspace tracking in impulsive noise," *IEEE Trans. Signal Process.*, vol. 54, no. 1, pp. 105–116, 2006.

[159] V.-D. Nguyen, N. L. Trung, and K. Abed-Meraim, "Robust subspace tracking algorithms using fast adaptive Mahalanobis distance," *Signal Process.*, vol. 195, p. 108402, 2022.

[160] A. M. Rekavandi, A.-K. Seghouane, and K. Abed-Meraim, "TRPAST: A tunable and robust projection approximation subspace tracking method," *IEEE Trans. Signal Process.*, 2022 (submitted).

[161] L. T. Thanh, A. M. Rekavandi, S. Abd-Krim, and K. Abed-Meraim, "Robust subspace tracking with contamination via α-divergence," in *IEEE Int. Conf. Acoust. Speech Signal Process.*, 2023 (submitted).

[162] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: A comparative review," *J. Mach. Learn. Res.*, vol. 10, no. 66-71, p. 13, 2009.

[163] J.-M. Chaufray, W. Hachem, and P. Loubaton, "Asymptotic analysis of optimum and suboptimum CDMA downlink MMSE receivers," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2620–2638, 2004.

[164] X. Mestre and M. Á. Lagunas, "Modified subspace algorithms for DoA estimation with large arrays," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 598–614, 2008.

[165] T. T. Cai, C.-H. Zhang, and H. H. Zhou, "Optimal rates of convergence for covariance matrix estimation," *Ann. Stat.*, vol. 38, no. 4, pp. 2118–2144, 2010.

[166] A. J. Rothman, E. Levina, and J. Zhu, "Generalized thresholding of large covariance matrices," *J. Am. Stat. Assoc.*, vol. 104, no. 485, pp. 177–186, 2009.

[167] M. Hardt and E. Price, "The noisy power method: A meta algorithm with applications," in *Adv. Neural Inf. Process. Syst.*, vol. 27, 2014.

[168] L. W. Mackey, "Deflation methods for sparse PCA," in *Adv. Neural Inf. Process. Syst.*, 2008, pp. 1017–1024.

[169] X.-T. Yuan and T. Zhang, "Truncated Power Method for Sparse Eigenvalue Problems," *J. Mach. Learn. Res.*, vol. 14, no. Apr, pp. 899–925, 2013.

[170] Y. Deshpande and A. Montanari, "Sparse PCA via covariance thresholding," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 4913–4953, 2016.

[171] T. Wang, Q. Berthet, and R. J. Samworth, "Statistical and computational trade-offs in estimation of sparse principal components," *Ann. Stat.*, vol. 44, no. 5, pp. 1896–1930, 2016.

[172] R. Krauthgamer, B. Nadler, D. Vilenchik *et al.*, "Do semidefinite relaxations solve sparse PCA up to the information limit?" *Ann. Stat.*, vol. 43, no. 3, pp. 1300–1322, 2015.

[173] V. Q. Vu and J. Lei, "Minimax sparse principal subspace estimation in high dimensions," *Ann. Stat.*, vol. 41, no. 6, pp. 2905–2947, 2013.

[174] N. V. Dung, K. Abed-Meraim, and N. L. Trung, "Second-order optimization based adaptive PARAFAC decomposition of three-way tensors," *Digit. Signal Process.*, vol. 63, pp. 100–111, 2017.

[175] S. Zhou, N. X. Vinh, J. Bailey, Y. Jia, and I. Davidson, "Accelerating online CP decompositions for higher order tensors," in *ACM SIGKDD Int. Conf. Knowl. Discover. Data Min.*, 2016, pp. 1375–1384.

[176] H. Kasai, "Fast online low-rank tensor subspace tracking by CP decomposition using recursive least squares from incomplete observations," *Neurocomput.*, vol. 347, pp. 177–190, 2019.

[177] X.-W. Chang, "On the Perturbation of the Q-factor of the QR Factorization," *Numer. Linear Algebra Appl.*, vol. 19, no. 3, pp. 607–619, 2012.

[178] I. Mitliagkas, C. Caramanis, and P. Jain, "Memory limited, streaming PCA," in *Adv. Neural Inf. Process. Syst.*, 2013, pp. 2886–2894.

[179] L. T. Thanh, N. T. A. Dao, N. V. Dung, N. L. Trung, and K. Abed-Meraim, "Multi-channel EEG epileptic spike detection by a new method of tensor decomposition," *J. Neural Eng.*, vol. 17, no. 1, p. 016023, 2020.

[180] F. Cong, Q.-H. Lin, L.-D. Kuang, X.-F. Gong, P. Astikainen, and T. Ristaniemi, "Tensor decomposition of EEG signals: A brief review," *J. Neurosci. Methods*, vol. 248, pp. 59–69, 2015.

[181] D. Nion and N. D. Sidiropoulos, "Tensor algebra and multidimensional harmonic retrieval in signal processing for MIMO radar," *IEEE Trans. Signal Process.*, vol. 58, no. 11, pp. 5693–5705, 2010.

[182] H. Chen, F. Ahmad, S. Vorobyov, and F. Porikli, "Tensor decompositions in wireless communications and MIMO radar," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 3, pp. 438–453, 2021.

[183] M. Nakatsuji, Q. Zhang, X. Lu, B. Makni, and J. A. Hendler, "Semantic social network analysis by cross-domain tensor factorization," *IEEE Trans. Comput. Soc. Syst.*, vol. 4, no. 4, pp. 207–217, 2017.

[184] S. Fernandes, H. Fanaee-T, and J. Gama, "Tensor decomposition for analysing time-evolving social networks: An overview," *Artif. Intell. Rev.*, vol. 54, no. 4, pp. 2891–2916, 2021.

[185] R. Bro, "PARAFAC. Tutorial and applications," *Chemometr. Intell. Lab. Syst.*, vol. 38, no. 2, pp. 149–172, 1997.

[186] E. Acar and B. Yener, "Unsupervised multiway data analysis: A literature survey," *IEEE Trans Knowl. Data Eng.*, vol. 21, no. 1, pp. 6–20, 2008.

[187] G. Bergqvist and E. G. Larsson, "The higher-order singular value decomposition: Theory and an application," *IEEE Signal Process. Mag.*, vol. 27, no. 3, pp. 151–154, 2010.

[188] L. Grasedyck, D. Kressner, and C. Tobler, "A literature survey of low-rank tensor approximation techniques," *GAMM-Mitteilungen*, vol. 36, no. 1, pp. 53–78, 2013.

[189] N. Vervliet, O. Debals, L. Sorber, and L. De Lathauwer, "Breaking the Curse of Dimensionality Using Decompositions of Incomplete Tensors: Tensor-based scientific computing in big data analysis," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 71–79, 2014.

[190] V. D. Nguyen, K. Abed-Meraim, and N. Linh-Trung, "Fast tensor decompositions for big data processing," in *Int. Conf. Adv. Technol. Commun.*, 2016, pp. 215–221.

[191] S. A. Asl, A. Cichocki, A. H. Phan, I. Oseledets *et al.*, "Randomized algorithms for computation of Tucker decomposition and higher-order SVD (HOSVD)," *IEEE Access*, vol. 9, pp. 28 684–28 706, 2021.

[192] X. Fu, N. Vervliet, L. De Lathauwer, K. Huang, and N. Gillis, "Computing large-scale matrix and tensor decomposition with structured factors: A unified nonconvex optimization perspective," *IEEE Signal Process. Mag.*, vol. 37, no. 5, pp. 78–94, 2020.

[193] D. Muti and S. Bourennane, "Survey on tensor signal algebraic filtering," *Signal Process.*, vol. 87, no. 2, pp. 237–249, 2007.

[194] P. Comon, X. Luciani, and A. L. De Almeida, "Tensor decompositions, alternating least squares and other tales," *J. Chemom.*, vol. 23, no. 7-8, pp. 393–405, 2009.

[195] C. J. Hillar and L.-H. Lim, "Most tensor problems are NP-hard," *J. ACM*, vol. 60, no. 6, p. 45, 2013.

[196] P. Comon, "Tensors : A brief introduction," *IEEE Signal Process. Mag.*, vol. 31, no. 3, pp. 44–53, 2014.

[197] A. Zare, A. Ozdemir, M. A. Iwen, and S. Aviyente, "Extension of PCA to higher order data structures: An introduction to tensors, tensor decompositions, and tensor PCA," *Proc. IEEE*, vol. 106, no. 8, pp. 1341–1358, 2018.

[198] M. Mørup, "Applications of tensor (multiway array) factorizations and decompositions in data mining," *Data Min. Knowl. Discov.*, vol. 1, no. 1, pp. 24–40, 2011.

[199] A. Cichocki, D. Mandic, L. D. Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor Decompositions for Signal Processing Applications: From two-way to multiway component analysis," *IEEE Signal Process. Mag.*, vol. 32, no. 2, pp. 145–163, 2015.

[200] H. Fanaee-T and J. Gama, "Tensor-based anomaly detection: An interdisciplinary survey," *Knowl. Based Syst.*, vol. 98, pp. 130–147, 2016.

[201] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Tensors for data mining and data fusion: Models, applications, and scalable algorithms," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 2, p. 16, 2017.

[202] A. Cichocki, A.-H. Phan, Q. Zhao, N. Lee, I. Oseledets, M. Sugiyama, and D. P. Mandic, "Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives," *Found. Trends Mach. Learn.*, vol. 9, no. 6, pp. 431–673, 2017.

[203] Y. Ji, Q. Wang, X. Li, and J. Liu, "A survey on tensor techniques and applications in machine learning," *IEEE Access*, vol. 7, pp. 162 950–162 990, 2019.

[204] S. Miron, Y. Zniyed, R. Boyer, A. De Almeida, G. Favier, D. Brie, and P. Comon, "Tensor methods for multisensor signal processing," *IET Signal Process.*, vol. 14, no. 10, pp. 693–709, 2021.

[205] Y. Panagakis, J. Kossaifi, G. G. Chrysos, J. Oldfield, M. A. Nicolaou, A. Anandkumar, and S. Zafeiriou, "Tensor methods in computer vision and deep learning," *Proc. IEEE*, vol. 109, no. 5, pp. 863–890, 2021.

[206] K. Batselier, "Low-rank tensor decompositions for nonlinear system identification: A tutorial with examples," *IEEE Control Syst. Mag.*, vol. 42, no. 1, pp. 54–74, 2022.

[207] V. De Silva and L.-H. Lim, "Tensor rank and the ill-posedness of the best low-rank approximation problem," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1084–1127, 2008.

[208] L. De Lathauwer, B. De Moor, and J. Vandewalle, "On the best rank-1 and rank-(r_1, r_2, r_n) approximation of higher-order tensors," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1324–1342, 2000.

[209] L. De Lathauwer and D. Nion, "Decompositions of a higher-order tensor in block terms—Part III: Alternating least squares algorithms," *SIAM J. Matrix Anal. Appl.*, vol. 30, pp. 1067–1083, 2008.

[210] H. Fanaee-T and J. Gama, "Multi-aspect-streaming tensor analysis," *Knowl. Based Syst.*, vol. 89, pp. 332–345, 2015.

[211] D. Nion and N. D. Sidiropoulos, "Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor," *IEEE Trans. Signal Process.*, vol. 57, no. 6, pp. 2299–2310, 2009.

[212] V. D. Nguyen, K. Abed-Meraim, and N. Linh-Trung, "Fast adaptive PARAFAC decomposition algorithm with linear complexity," in *IEEE Int. Conf. Acoust. Speech Signal Process.*, 2016, pp. 6235–6239.

[213] M. Vandecappelle, N. Vervliet, and L. De Lathauwer, "Nonlinear least squares updating of the canonical polyadic decomposition," in *Eur. Signal Process. Conf.*, 2017, pp. 663–667.

[214] Z. Zhang and C. Hawkins, "Variational bayesian inference for robust streaming tensor factorization and completion," in *IEEE Int. Conf. Data Min.*, 2018, pp. 1446–1451.

[215] T. Minh-Chinh, V. D. Nguyen, N. Linh-Trung, and K. Abed-Meraim, "Adaptive PARAFAC decomposition for third-order tensor completion," in *IEEE Int. Conf. Commun. Elect.*, 2016, pp. 297–301.

[216] S. Smith, K. Huang, N. D. Sidiropoulos, and G. Karypis, "Streaming tensor factorization for infinite data sources," in *SIAM Int. Conf. Data Min.*, 2018, pp. 81–89.

[217] Y. Du, Y. Zheng, K. Lee, and S. Zhe, "Probabilistic streaming tensor decomposition," in *IEEE Int. Conf. Data Min.*, 2018, pp. 99–108.

[218] H.-K. Yang and H.-S. Yong, "Incremental PARAFAC decomposition for three-dimensional tensors using Apache Spark," in *Int. Conf. Web Eng.*, 2019, pp. 63–71.

[219] S. Rambhatla, X. Li, and J. Haupt, "Provable online CP /PARAFAC decomposition of a structured tensor via dictionary learning," in *Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1–12.

[220] E. Gujral, G. Theocharous, and E. E. Papalexakis, "SPADE: Streaming PARAFAC2 decomposition for large datasets," in *SIAM Int. Conf. Data Min.*, 2020, pp. 577–585.

[221] T. Kwon, I. Park, D. Lee, and K. Shin, "SliceNStitch: Continuous CP decomposition of sparse tensor streams," *IEEE Int. Conf. Data Eng.*, pp. 816–827, 2021.

[222] L. Dongjin and S. Kijung, "Robust factorization of real-world tensor streams with patterns, missing values, and outliers," in *IEEE Int. Conf. Data Eng.*, 2021, pp. 840–851.

[223] D. Ahn, S. Kim, and U. Kang, "Accurate online tensor factorization for temporal tensor streams with missing values," in *ACM Int. Conf. Inf. Knowl. Manag.*, 2021, pp. 2822–2826.

[224] H. Lyu, C. Strohmeier, and D. Needell, "Online nonnegative CP-dictionary learning for Markovian data," *J. Mach. Learn. Res.*, pp. 1–41, 2022 (to appear).

[225] R. A. Harshman, "PARAFAC2: Mathematical and technical notes," *UCLA Work. Pap. Phon.*, vol. 22, pp. 30–44, 1972.

[226] C. Chatfield, "The holt-winters forecasting procedure," *J. R. Stat. Soc. Ser. C Appl. Stat.*, vol. 27, no. 3, pp. 264–279, 1978.

[227] P. Strobach, "Bi-iteration SVD subspace tracking algorithms," *IEEE Trans. Signal Process.*, vol. 45, no. 5, pp. 1222–1240, 1997.

[228] C. Zeng and M. K. Ng, "Incremental CP tensor decomposition by alternating minimization method," *SIAM J. Matrix Anal. Appl.*, vol. 42, no. 2, pp. 832–858, 2021.

[229] S. Fang, Z. Wang, Z. Pan, J. Liu, and S. Zhe, "Streaming Bayesian deep tensor factorization," in *Int. Conf. Machine Learn.*, 2021, pp. 3133–3142.

[230] T. Broderick, N. Boyd, A. Wibisono, A. C. Wilson, and M. I. Jordan, "Streaming variational bayes," in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26, 2013.

[231] X. Boyen and D. Koller, "Tractable inference for complex stochastic processes," in *Conf. Uncertain. Artif. Intell.*, 1998, pp. 33–42.

[232] Q. Song, X. Huang, H. Ge, J. Caverlee, and X. Hu, "Multi-aspect streaming tensor completion," in *ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2017, pp. 435–443.

[233] M. Najafi, L. He, and S. Y. Philip, "Outlier-robust multi-aspect streaming tensor completion and factorization." in *IJCAI Int. Joint Conf. Artif. Intell.*, 2019, pp. 3187–3194.

[234] H.-K. Yang and H.-S. Yong, "Multi-aspect incremental tensor decomposition based on distributed in-memory big data systems," *J. Data Inf. Sci.*, vol. 5, no. 2, pp. 13–32, 2020.

[235] K. Yang, Y. Gao, Y. Shen, B. Zheng, and L. Chen, "DisMASTD: An efficient distributed multi-aspect streaming tensor decomposition," in *IEEE Int. Conf. Data Eng.*, 2021, pp. 1080–1091.

[236] J. Sun, D. Tao, and C. Faloutsos, "Beyond streams and graphs: Dynamic tensor analysis," in *ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2006, pp. 374–383.

[237] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos, "Incremental tensor analysis: Theory and applications," *ACM Trans. Knowl. Discov. Data*, vol. 2, no. 3, pp. 1–37, 2008.

[238] X. Li, W. Hu, Z. Zhang, X. Zhang, and G. Luo, "Robust visual tracking based on incremental tensor subspace learning," in *IEEE Int. Conf. Comput. Vis.*, 2007, pp. 1–8.

[239] 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, no. 3, pp. 303–327, 2011.

[240] W. Zhang, H. Sun, X. Liu, Xiaohui, and Guo, "An incremental tensor factorization approach for web service recommendation," in *IEEE Int. Conf. Data Min. Works.*, 2014, pp. 346–351.

[241] L. Kuang, F. Hao, L. T. Yang, M. Lin, C. Luo, and G. Min, "A tensor-based approach for big data representation and dimensionality reduction," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 3, pp. 280–291, 2014.

[242] R. Yu, D. Cheng, and Y. Liu, "Accelerated online low rank tensor learning for multivariate spatiotemporal streams," in *Int. Conf. Mach. Learn.*, 2015, pp. 238–247.

[243] M. Baskaran, M. H. Langston, T. Ramananandro, D. Bruns-Smith, T. Henretty, J. Ezick, and R. Lethin, "Accelerated low-rank updates to

tensor decompositions," in *IEEE High Perf. Extreme Comput. Conf.*, 2016, pp. 1–7.

[244] H. Kasai and B. Mishra, "Low-rank tensor completion: A Riemannian manifold preconditioning approach," in *Int. Conf. Mach. Learn.*, 2016, pp. 1012–1021.

[245] A. Ozdemir, E. M. Bernat, and S. Aviyente, "Recursive tensor subspace tracking for dynamic brain network analysis," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 4, pp. 669–682, 2017.

[246] X. Wang, W. Wang, L. T. Yang, S. Liao, D. Yin, and M. J. Deen, "A distributed HOSVD method with its incremental computation for big data in cyber-physical-social systems," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 2, pp. 481–492, 2018.

[247] L. T. Yang, X. Wang, X. Chen, L. Wang, R. Ranjan, X. Chen, and M. J. Deen, "A multi-order distributed HOSVD with its incremental computing for big services in cyber-physical-social systems," *IEEE Trans. Big Data*, vol. 6, no. 4, pp. 666–678, 2020.

[248] N. Madhav, B. Mishra, M. Gupta, and P. Talukdar, "Inductive framework for multi-aspect streaming tensor completion with side information," in *ACM Int. Conf. Inf. Knowl. Manag.*, 2018, pp. 307–316.

[249] H. Xiao, F. Wang, F. Ma, and J. Gao, "eOTD: An efficient online tucker decomposition for higher order tensors," in *IEEE Int. Conf. Data Min.*, 2018, pp. 1326–1331.

[250] A. Traoré, M. Berar, and A. Rakotomamonjy, "Online multimodal dictionary learning," *Neurocomput.*, vol. 368, pp. 163–179, 2019.

[251] ——, "Singleshot: A scalable Tucker tensor decomposition," in *Adv. Neural Inf. Process. Syst.*, 2019.

[252] 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.

[253] Z. Pan, Z. Wang, and S. Zhe, "Streaming nonlinear Bayesian tensor decomposition," in *Conf. Uncertain. Artif. Intell.*, 2020, pp. 490–499.

[254] D. G. Chachlakis, M. Dhanaraj, A. Prater-Bennette, and P. P. Markopoulos, "Dynamic L1-norm Tucker tensor decomposition," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 3, pp. 587–602, 2021.

[255] S. Fang, R. M. Kirby, and S. Zhe, "Bayesian streaming sparse Tucker decomposition," in *Conf. Uncertain. Artif. Intell.*, 2021, pp. 558–567.

[256] J.-G. Jang and U. Kang, "Fast and memory-efficient tucker decomposition for answering diverse time range queries," in *ACM SIGKDD Conf. Knowl. Discov. Data Min.*, 2021, pp. 725–735.

[257] R. Zdunek and K. Fonal, "Incremental nonnegative Tucker decomposition with block-coordinate descent and recursive approaches," *Symmetry*, vol. 14, no. 1, p. 113, 2022.

[258] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, no. 1, pp. 125–141, 2008.

[259] J. Li, G. Han, J. Wen, and X. Gao, "Robust tensor subspace learning for anomaly detection," *Int. J. Mach. Learn. Cybern.*, vol. 2, no. 2, pp. 89–98, 2011.

[260] X. Wang, L. T. Yang, X. Chen, M. J. Deen, and J. Jin, "Improved multiorder distributed HOSVD with its incremental computing for smart city services," *IEEE Trans. Sust. Comput.*, vol. 6, no. 3, pp. 456–468, 2021.

[261] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "A survey of multilinear subspace learning for tensor data," *Pattern Recognit.*, vol. 44, no. 7, pp. 1540–1551, 2011.

[262] R. Zhao and Q. Wang, "Learning separable dictionaries for sparse tensor representation: An online approach," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 66, no. 3, pp. 502–506, 2019.

[263] P. Li, J. Feng, X. Jin, L. Zhang, X. Xu, and S. Yan, "Online robust low-rank tensor modeling for streaming data analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1061–1075, 2019.

[264] Y. Hu, A. Qu, Y. Wang, and D. Work, "Streaming data preprocessing via online tensor recovery for large environmental sensor networks," *ArXiv Prepr. ArXiv210900596*, 2021.

[265] D. G. Chachlakis, A. Prater-Bennette, and P. P. Markopoulos, "L1-norm Tucker tensor decomposition," *IEEE Access*, vol. 7, pp. 178 454–178 465, 2019.

[266] D. Kressner, M. Steinlechner, and B. Vandereycken, "Low-rank tensor completion by Riemannian optimization," *BIT Numer. Math.*, vol. 54, no. 2, pp. 447–468, 2014.

[267] H. Liu, L. T. Yang, Y. Guo, X. Xie, and J. Ma, "An incremental tensor-train decomposition for cyber-physical-social big data," *IEEE Trans. Big Data*, vol. 7, no. 2, pp. 341–354, 2021.

[268] X. Wang, L. T. Yang, Y. Wang, L. Ren, and M. J. Deen, "ADTT: A highly efficient distributed tensor-train decomposition method for IIoT big data," *IEEE Trans Ind. Inf.*, vol. 17, no. 3, pp. 1573–1582, 2021.

[269] E. Gujral and E. E. Papalexakis, "OnlineBTD: Streaming algorithms to track the block term decomposition of large tensors," in *IEEE Int. Conf. Data Sci. Adv. Anal.*, 2020, pp. 168–177.

[270] A. A. Rontogiannis, E. Kofidis, and P. V. Giampouras, "Online rank-revealing block-term tensor decomposition," in *Asilomar Conf. Signals Syst. Comput.*, 2021, pp. 1678–1682.

[271] Z. Zhang, D. Liu, S. Aeron, and A. Vetro, "An online tensor robust PCA algorithm for sequential 2D data," in *IEEE Int. Conf. Acoust. Speech Signal Process.*, 2016, pp. 2434–2438.

[272] K. Gilman, D. Ataee Tarzanagh, and L. Balzano, "Grassmannian Optimization for Online Tensor Completion and Tracking with the t-SVD," *IEEE Trans. Signal Process.*, pp. 1–1, 2022.

[273] K. Gilman and L. Balzano, "Online tensor completion and free submodule tracking with the t-SVD," in *IEEE Int. Conf. Acoust. Speech Signal Process.*, 2020, pp. 3282–3286.

[274] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, 2013.

[275] X. Zhang, X. Shi, W. Hu, X. Li, and S. Maybank, "Visual tracking via dynamic tensor analysis with mean update," *Neurocomput.*, vol. 74, no. 17, pp. 3277–3285, 2011.

[276] W. Hu, J. Gao, J. Xing, C. Zhang, and S. Maybank, "Semi-supervised tensor-based graph embedding learning and its application to visual discriminant tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 1, pp. 172–188, 2016.

[277] S. Khan, G. Xu, R. Chan, and H. Yan, "An online spatio-temporal tensor learning model for visual tracking and its applications to facial expression recognition," *Expert Syst. Appl.*, vol. 90, pp. 427–438, 2017.

[278] A. Sobral, S. Javed, S. K. Jung, T. Bouwmans, and E. Zahzah, "Online stochastic tensor decomposition for background subtraction in multispectral video sequences," in *IEEE Int. Conf. Comput. Vis.*, 2015, pp. 946–953.

[279] M. M. Salut and D. V. Anderson, "Online tensor robust principal component analysis," *IEEE Access*, vol. 10, pp. 69 354–69 363, 2022.

[280] Y. He and G. K. Atia, "Patch tracking-based streaming tensor ring completion for visual data recovery," *IEEE Trans. Circuits Syst. Video Techn.*, pp. 1–1, 2022.

[281] B. Wen, Y. Li, L. Pfister, and Y. Bresler, "Joint adaptive sparsity and low-rankness on the fly: an online tensor reconstruction scheme for video denoising," in *IEEE Int. Conf. Comput. Vis.*, 2017, pp. 241–250.

[282] B. Wen, S. Ravishankar, and Y. Bresler, "VIDOSAT: High-dimensional sparsifying transform learning for online video denoising," *IEEE Trans. Image Process.*, vol. 28, no. 4, pp. 1691–1704, 2018.

[283] C. Min and G. Medioni, "Inferring segmented dense motion layers using 5D tensor voting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 9, pp. 1589–1602, 2008.

[284] D. S. Bassett and M. S. Gazzaniga, "Understanding complexity in the human brain," *Trends Cognitive Sci.*, vol. 15, no. 5, pp. 200–209, 2011.

[285] A. Cichocki, Y. Washizawa, T. Rutkowski, H. Bakardjian, A.-H. Phan, S. Choi, H. Lee, Q. Zhao, L. Zhang, and Y. Li, "Noninvasive BCIs: Multiway signal-processing array decompositions," *Computer*, vol. 41, no. 10, pp. 34–42, 2008.

[286] N. Yeung, M. M. Botvinick, and J. D. Cohen, "The neural basis of error detection: conflict monitoring and the error-related negativity." *Psychological Rev.*, vol. 111, no. 4, p. 931, 2004.

[287] A. G. Mahyari, D. M. Zoltowski, E. M. Bernat, and S. Aviyente, "A tensor decomposition-based approach for detecting dynamic network states from EEG," *IEEE Trans. Biomed. Eng.*, vol. 64, no. 1, pp. 225–237, 2016.

[288] E. Acar, M. Roald, K. M. Hossain, V. D. Calhoun, and T. Adali, "Tracing evolving networks using tensor factorizations vs. ICA-based approaches," *Front. Neurosci.*, vol. 16, 2022.

[289] A. Fotouhi, E. Eqlimi, and B. Makkiabadi, "Evaluation of adaptive PARAFAC alogorithms for tracking of simulated moving brain sources,"

in *Annual Int. Conf. IEEE Eng. Med. Biol. Society.*   IEEE, 2015, pp. 3819–3822.

[290] J. W. Meijs, O. W. Weier, M. J. Peters, and A. Van Oosterom, "On the numerical accuracy of the boundary element method (EEG application)," *IEEE Trans. Biomedical Eng.*, vol. 36, no. 10, pp. 1038–1049, 1989.

[291] A. Fotouhi, E. Eqlimi, and B. Makkiabadi, "Adaptive localization of moving eeg sources using augmented complex tensor factorization," in *IEEE Int. Conf. Telecommun. Signal Process.*   IEEE, 2017, pp. 439–443.

[292] N. Linh-Trung, T. Minh-Chinh, V. Nguyen, and K. Abed-Meraim, "A non-linear tensor tracking algorithm for analysis of incomplete multi-channel EEG data," in *Int. Symp. Med. Inf. Commun. Technol.*, 2018, pp. 1–6.

[293] E. Karahan, P. A. Rojas-Lopez, M. L. Bringas-Vega, P. A. Valdes-Hernandez, and P. A. Valdes-Sosa, "Tensor analysis and fusion of multimodal brain images," *Proc. IEEE*, vol. 103, no. 9, pp. 1531–1559, 2015.

[294] G. Zhou, Q. Zhao, Y. Zhang, T. Adalı, S. Xie, and A. Cichocki, "Linked component analysis from matrices to high-order tensors: Applications to biomedical data," *Proceed. IEEE*, vol. 104, no. 2, pp. 310–331, 2016.

[295] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.

[296] L. Shi, A. Gangopadhyay, and V. P. Janeja, "STenSr: Spatio-temporal tensor streams for anomaly detection and pattern discovery," *Knowl. Inf. Syst.*, vol. 43, no. 2, pp. 333–353, 2015.

[297] H. Kasai, W. Kellerer, and M. Kleinsteuber, "Network volume anomaly detection and identification in large-scale networks based on online time-structured traffic tensor tracking," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 636–650, 2016.

[298] N. Cao, C. Lin, Q. Zhu, Y.-R. Lin, X. Teng, and X. Wen, "Voila: Visual anomaly detection and monitoring with streaming spatiotemporal data," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 1, pp. 23–33, 2017.

[299] C. Lin, Q. Zhu, S. Guo, Z. Jin, Y.-R. Lin, and N. Cao, "Anomaly detection in spatiotemporal data via regularized non-negative tensor analysis," *Data Min. Knowl. Disc.*, vol. 32, no. 4, pp. 1056–1073, 2018.

[300] M. Xu, J. Wu, H. Wang, and M. Cao, "Anomaly detection in road networks using sliding-window tensor factorization," *IEEE Trans. Intell. Transport. Syst.*, vol. 20, no. 12, pp. 4704–4713, 2019.

[301] J. Yuan, G. C. Alexandropoulos, E. Kofidis, T. L. Jensen, and E. De Carvalho, "Channel tracking for ris-enabled multi-user simo systems in time-varying wireless channels," in *IEEE Int. Conf. Commun. Works.* IEEE, 2022, pp. 145–150.

[302] K. Luo, X. Zhou, B. Wang, J. Huang, and H. Liu, "Sparse Bayes tensor and DOA tracking inspired channel estimation for V2X millimeter wave massive MIMO system," *Sensors*, vol. 21, no. 12, p. 4021, 2021.

[303] C. C. Garcez, D. V. de Lima, R. K. Miranda, F. Mendonca, J. P. C. da Costa, A. L. de Almeida, and R. T. de Sousa Jr, "Tensor-based subspace tracking for time-delay estimation in GNSS multi-antenna receivers," *Sensors*, vol. 19, no. 23, p. 5076, 2019.

[304] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *ACM Trans. Knowl. Disc. Data*, vol. 5, no. 2, pp. 1–27, 2011.

[305] Y.-R. Lin, K. S. Candan, H. Sundaram, and L. Xie, "SCENT: Scalable compressed monitoring of evolving multirelational social networks," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 7, no. 1, pp. 1–22, 2011.

[306] K. Shin, B. Hooi, J. Kim, and C. Faloutsos, "Densealert: Incremental dense-subtensor detection in tensor streams," in *ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2017, pp. 1057–1066.

[307] W. Sun and R. D. Braatz, "Opportunities in tensorial data analytics for chemical and biological manufacturing processes," *Comput. Chem. Eng.*, vol. 143, p. 107099, 2020.

[308] I. W. Sanou, R. Redon, X. Luciani, and S. Mounier, "Online Nonnegative and Sparse Canonical Polyadic Decomposition of Fluorescence Tensors," *Chemometrics and Intelligent Laboratory Systems*, p. 104550, 2022.

[309] X. Meng, A. Morris, and E. Martin, "On-line monitoring of batch processes using a PARAFAC representation," *J. Chemometr.*, vol. 17, no. 1, pp. 65–81, 2003.

[310] S. Gourvenec, I. Stanimirova, C.-A. Saby, C. Airiau, and D. Massart, "Monitoring batch processes with the STATIS approach," *J. Chemometr.*, vol. 19, no. 5-7, pp. 288–300, 2005.

[311] H. Tan, Y. Wu, B. Shen, P. J. Jin, and B. Ran, "Short-term traffic prediction based on dynamic tensor completion," *IEEE Trans. Intell. Transport. Syst.*, vol. 17, no. 8, pp. 2123–2133, 2016.

[312] J. Wang, J. Wu, Z. Wang, F. Gao, and Z. Xiong, "Understanding urban dynamics via context-aware tensor factorization with neighboring regularization," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 11, pp. 2269–2283, 2019.

[313] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 171–209, 2014.

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

[315] C. D. Martin, R. Shafer, and B. LaRue, "An order-p tensor factorization with applications in imaging," *SIAM J. Sci. Comput.*, vol. 35, no. 1, pp. A474–A490, 2013.

[316] Z. Zhang and S. Aeron, "Exact tensor completion using t-SVD," *IEEE Trans. Signal Process.*, vol. 65, no. 6, pp. 1511–1526, 2017.

[317] F. Jiang, X.-Y. Liu, H. Lu, and R. Shen, "Efficient multi-dimensional tensor sparse coding using t-linear combination," in *AAAI Conf. Artif. Intell.*, 2018.

[318] I. Kajo, N. Kamel, and Y. Ruichek, "Incremental tensor-based completion method for detection of stationary foreground objects," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 5, pp. 1325–1338, 2019.

[319] S. Chatterjee, "A deterministic theory of low rank matrix completion," *IEEE Trans. Inf. Theory*, vol. 66, no. 12, pp. 8046–8055, 2020.

[320] M. Ashraphijuo and X. Wang, "Fundamental conditions for low-CP-rank tensor completion," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 2116–2145, 2017.

[321] J. Mairal, "Incremental majorization-minimization optimization with application to large-scale machine learning," *SIAM J. Optim.*, vol. 25, no. 2, pp. 829–855, 2015.

[322] G. Raskutti and M. W. Mahoney, "A statistical perspective on randomized sketching for ordinary least-squares," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 7508–7538, 2016.

[323] M. W. Mahoney, "Randomized algorithms for matrices and data," *Found. Trends Mach. Learn.*, vol. 3, no. 2, pp. 123–224, 2011.

[324] C. Battaglino, G. Ballard, and T. G. Kolda, "A practical randomized CP tensor decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 39, no. 2, pp. 876–901, 2018.

[325] H. Avron, P. Maymounkov, and S. Toledo, "Blendenpik: Supercharging LAPACK's least-squares solver," *SIAM J. Sci. Comput.*, vol. 32, no. 3, pp. 1217–1236, 2010.

[326] I. C. Ipsen and T. Wentworth, "The effect of coherence on sampling from matrices with orthonormal columns, and preconditioned least squares problems," *SIAM J. Matrix Anal. Appl.*, vol. 35, no. 4, pp. 1490–1520, 2014.

[327] J. A. Tropp, "Improved analysis of the subsampled randomized Hadamard transform," *Adv. Adapt. Data Anal.*, vol. 3, no. 01n02, pp. 115–126, 2011.

[328] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós, "Faster least squares approximation," *Numer. Math.*, vol. 117, no. 2, pp. 219–249, 2011.

[329] J. C. Spall, *Introduction to Stochastic Search and Optimization*, 2005.

[330] A. N. Langville and W. J. Stewart, "The Kronecker product and stochastic automata networks," *J. Comput. Appl. Math.*, vol. 167, no. 2, pp. 429–447, 2004.

[331] C. F. Van Loan, "The ubiquitous Kronecker product," *J. Comput. Appl. Math.*, vol. 123, no. 1-2, pp. 85–100, 2000.

[332] H. Diao, R. Jayaram, Z. Song, W. Sun, and D. Woodruff, "Optimal sketching for Kronecker product regression and low rank approximation," in *Adv. Neural Inf. Process. Syst.*, 2019, pp. 4739–4750.

[333] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-minimization algorithms in signal processing, communications, and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 3, pp. 794–816, 2016.

[334] N. Guan, D. Tao, Z. Luo, and B. Yuan, "Online nonnegative matrix factorization with robust stochastic approximation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 7, pp. 1087–1099, 2012.

[335] B. W. Bader and T. G. Kolda, "Efficient MATLAB computations with sparse and factored tensors," *SIAM J. Sci. Comput.*, vol. 30, no. 1, pp. 205–231, 2008.

[336] A.-H. Phan, P. Tichavský, and A. Cichocki, "Fast alternating LS algorithms for high order CANDECOMP/PARAFAC tensor factorizations," *IEEE Trans. Signal Process.*, vol. 61, no. 19, pp. 4834–4846, 2013.

[337] J. S. Simonoff, *Smoothing Methods in Statistics*, 2012.

[338] A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*, 2009.

[339] D. Chen and R. J. Plemmons, "Nonnegativity constraints in numerical analysis," in *Symposium on the Birth of Numerical Analysis*, 2010, pp. 109–139.

[340] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, 1995.

[341] D. L. Pimentel-Alarcón, N. Boston, and R. D. Nowak, "A characterization of deterministic sampling patterns for low-rank matrix completion," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 4, pp. 623–636, 2016.

[342] M. Ashraphijuo, V. Aggarwal, and X. Wang, "Deterministic and probabilistic conditions for finite completability of low-Tucker-Rank tensor," *IEEE Trans. Inf. Theory*, vol. 65, no. 9, pp. 5380–5400, 2019.

[343] M. Métivier, *Semimartingales: A Course on Stochastic Processes*, 1984.

[344] Y. Xu, "Fast algorithms for higher-order singular value decomposition from incomplete data," *J. Comput. Math.*, vol. 35, no. 4, pp. 395–420, 2017.

[345] M. Filipović and A. Jukić, "Tucker factorization with missing data with application to low-n-rank tensor completion," *Multidimens. Syst. Signal Process.*, vol. 26, no. 3, pp. 677–692, 2015.

[346] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup, "Scalable tensor factorizations for incomplete data," *Chemometr. Intell. Lab. Syst.*, vol. 106, no. 1, pp. 41–56, 2011.

[347] V. Vigneron, A. Kodewitz, M. N. da Costa, A. M. Tome, and E. Langlang, "Non-negative sub-tensor ensemble factorization (NsTEF) algorithm. A new incremental tensor factorization for large datasets." *Signal Process.*, vol. 144, pp. 77–86, 2018.

[348] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, "Rank-one modification of the symmetric eigenproblem," *Numer. Math.*, vol. 31, no. 1, pp. 31–48, 1978.

[349] G. Cheng, X. Luo, and L. Li, "The bounds of the smallest and largest eigenvalues for rank-one modification of the Hermitian eigenvalue problem," *Applied Mathematics Letters*, vol. 25, no. 9, pp. 1191–1196, 2012.

[350] J. F. Bonnans and A. Shapiro, "Optimization problems with perturbations: A guided tour," *SIAM Rev.*, vol. 40, no. 2, pp. 228–264, 1998.

[351] J. Mairal, "Stochastic majorization-minimization algorithms for large-scale optimization," in *Adv. Neural Inf. Process. Syst.*, 2013, pp. 2283–2291.

[352] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2002.

[353] Y. Nesterov, "Introductory lectures on convex programming," 1998.

[354] E. Kofidis, "A tensor-based approach to joint channel Estimation/Data detection in flexible multicarrier MIMO systems," *IEEE Trans. Signal Process.*, vol. 68, pp. 3179–3193, 2020.

[355] V. A. Miguel, J. E. Cohen, R. Cabral Farias, J. Chanussot, and P. Comon, "Nonnegative tensor CP decomposition of hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 5, pp. 2577–2588, 2016.

[356] S. Velasco-Forero and J. Angulo, "Classification of hyperspectral images by tensor modeling and additive morphological decomposition," *Pattern Recognit.*, vol. 46, no. 2, pp. 566–577, 2013.

[357] E. Acar, C. Aykut-Bingol, H. Bingol, R. Bro, and B. Yener, "Multiway analysis of epilepsy tensors," *Bioinformatics*, vol. 23, no. 13, pp. i10–i18, 2007.

[358] J. Hastad, "Tensor rank is NP-complete," *J. Algorithm.*, vol. 11, no. 4, pp. 644–654, 1990.

[359] R. J. Little and D. B. Rubin, *Statistical Analysis with Missing Data*, 2019.

[360] C. Lubich, T. Rohwedder, R. Schneider, and B. Vandereycken, "Dynamical approximation by hierarchical Tucker and tensor-train tensors," *SIAM J. Matrix Anal. Appl.*, vol. 34, no. 2, pp. 470–494, 2013.

[361] C. Lubich, I. V. Oseledets, and B. Vandereycken, "Time integration of tensor trains," *SIAM J. Numer. Anal.*, vol. 53, no. 2, pp. 917–941, 2015.

[362] C. Lubich, B. Vandereycken, and H. Walach, "Time integration of rank-constrained Tucker tensors," *SIAM J. Numer. Anal.*, vol. 56, no. 3, pp. 1273–1290, 2018.

[363] Y. Zniyed, R. Boyer, A. de Almeida, and G. Favier, "A TT-Based hierarchical framework for decomposing high-order tensors," *SIAM J. Sci. Comput.*, vol. 42, no. 2, pp. 822–848, 2020.

[364] S. S. Haykin, *Adaptive Filter Theory*, 2008.

[365] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM J. Imaging Sci.*, vol. 6, no. 3, pp. 1758–1789, 2013.

[366] J. Gorski, F. Pfeuffer, and K. Klamroth, "Biconvex sets and optimization with biconvex functions: A survey and extensions," *Math. Methods Oper. Res.*, vol. 66, no. 3, pp. 373–407, 2007.

[367] A. Bifet, *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*, 2010.

[368] A.-A. Saucan, T. Chonavel, C. Sintes, and J.-M. Le Caillec, "CPHD-DOA Tracking of multiple extended sonar targets in impulsive environments," *IEEE Trans. Signal Process.*, vol. 64, no. 5, pp. 1147–1160, 2016.

[369] S. Wang, Z. He, K. Niu, P. Chen, and Y. Rong, "New results on joint channel and impulsive noise estimation and tracking in underwater acoustic OFDM systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2601–2612, 2020.

[370] R. L. Das and M. Narwaria, "Lorentzian based adaptive filters for impulsive noise environments," *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 64, no. 6, pp. 1529–1539, 2017.

[371] P. Hänggi and P. Jung, "Colored noise in dynamical systems," *Adv. Chem. Phys.*, vol. 89, pp. 239–326, 2007.

[372] M. Mørup and L. K. Hansen, "Automatic relevance determination for multi-way models," *J. Chemom.*, vol. 23, no. 7-8, pp. 352–363, 2009.

[373] J. A. Bazerque, G. Mateos, and G. B. Giannakis, "Rank regularization and Bayesian inference for tensor completion and extrapolation," *IEEE Trans. Signal Process.*, vol. 61, no. 22, pp. 5689–5703, 2013.

[374] Q. Zhao, L. Zhang, and A. Cichocki, "Bayesian CP factorization of incomplete tensors with automatic rank determination," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1751–1763, 2015.

[375] M. Che, A. Cichocki, and Y. Wei, "Neural networks for computing best rank-one approximations of tensors and its applications," *Neurocomputing*, vol. 267, pp. 114–133, 2017.

[376] M. Zhou, Y. Liu, Z. Long, L. Chen, and C. Zhu, "Tensor rank learning in CP decomposition via convolutional neural network," *Signal Process. Image Commun.*, vol. 73, pp. 12–21, 2019.

[377] C. Hawkins, X. Liu, and Z. Zhang, "Towards compact neural networks via end-to-end training: A Bayesian tensor approach with automatic rank determination," *SIAM J. Math. Data Sci.*, vol. 4, no. 1, pp. 46–71, 2022.

[378] C. Ma, X. Yang, and H. Wang, "Randomized online CP decomposition," in *Int. Conf. Adv. Comput. Intell.*, 2018, pp. 414–419.

[379] I. Foster, *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*, 2020.

[380] J. H. Choi and S. Vishwanathan, "DFacTo: Distributed factorization of tensors," in *Adv. Neural Inf. Process. Syst.*, 2014, pp. 1–9.

[381] S. Smith, N. Ravindran, N. D. Sidiropoulos, and G. Karypis, "SPLATT: Efficient and parallel sparse tensor-matrix multiplication," in *IEEE Int. Parallel Distrib. Process. Symp.*, 2015, pp. 61–70.

[382] H. Li, Z. Li, K. Li, J. S. Rellermeyer, L. Chen, and K. Li, "SGD-Tucker: A novel stochastic optimization strategy for parallel sparse Tucker decomposition," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1828–1841, 2021.

[383] H. Al Daas, G. Ballard, P. Benner, and M. P. I. Magdeburg, "Parallel Algorithms for Tensor Train Arithmetic," *SIAM J. Sci. Comput.*, vol. 44, no. 1, pp. 25–53, 2022.

[384] N. Cohen and A. Shashua, "Convolutional rectifier networks as generalized tensor decompositions," in *Int. Conf. Mach. Learn.*, 2016, pp. 955–963.

[385] B. Liu, L. He, Y. Li, S. Zhe, and Z. Xu, "NeuralCP: Bayesian multiway data analysis with neural tensor decomposition," *Cogn. Comput.*, vol. 10, no. 6, pp. 1051–1061, 2018.

[386] X. Wang, M. Che, and Y. Wei, "Tensor neural network models for tensor singular value decompositions," *Comput. Optim. Appl.*, vol. 75, no. 3, pp. 753–777, 2020.

[387] P. Brito, "Symbolic data analysis: Another look at the interaction of data mining and statistics," *Data Min. Knowl. Discov.*, vol. 4, no. 4, pp. 281–295, 2014.

[388] F. Di Mauro, K. S. Candan, and M. L. Sapino, "Tensor-train decomposition in presence of interval-valued data," *IEEE Trans. Knowl. Data Eng.*, 2021 (early access).