A Non-Linear Tensor Tracking Algorithm for Analysis of Incomplete Multi-Channel EEG Data

Nguyen Linh-Trung[1], Truong Minh-Chinh[1,2], Viet-Dung Nguyen[1,3], Karim Abed-Meraim[4]
[1] AVITECH Institute, University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam
[2] Department of Physics, Hue University of Education, Hue, Vietnam
[3] L2S Laboratory, CentraleSupelec, University Paris-Saclay, Gif-sur-Yvette, France
[4] PRISME Laboratory, University of Orléans, Orléans, France

***Abstract*—Tensor decomposition is a popular tool to analyse and process data which can be represented by a higher-order tensor structure. In this paper, we consider tensor tracking in challenging situations where the observed data are streaming and incomplete. Specifically, we proposed a non-linear formulation of the PETRELS cost function and based on which we proposed NL-PETRELS subspace and tensor tracking algorithms. The non-linear function allows us to improve the convergence rate. We also illustrated the use of our proposed tensor tracking for incomplete multi-channel electroencephalogram (EEG) data in a real-life experiment in which the data can be represented by a third-order tensor.**

I. Introduction

Tensor decomposition is a popular tool to analyse and process data which can be represented by a higher-order tensor structure [1], [2]. In this paper, we are interested in using tensor decomposition in challenging situations where observed data are either *streaming* [3], [4] and/or *incomplete* [5]–[7].

Incomplete (missing, partial) observation of data occurs when we passively acquire the data partially, or when it is difficult or impossible to acquire all information. It also occurs when we actively schedule to acquire only a certain fraction of data, because of limitation in power consumption, storage and/or computational complexity. In such cases, the percentage of observed data can be moderate to very low, making classical processing approaches difficult to handle. Moreover, when data are of streaming (online) nature, processing them often requires fast updating instead of recalculating from the beginning due to time constraints.

In this paper, we are also interested in the use of tensor decomposition for a special type of data– electroencephalography (EEG). EEG records the electrical activity of the brain via electrodes adhered to the scalp [8]. EEG is used for diagnosis and treatment of various brain disorders, for example localizing the lesion in the brain that causes an epileptic seizure. Tensor decomposition has been shown to successfully represent and analyse EEG signals [9]–[12]. The reason for the success is that EEG signals are multi-dimensional while tensors provide a natural representation of multi-dimensional signals. Each single-channel EEG signal (i.e., recorded from one electrode) is a record in time of the brain activity, and thus provides a dimension of time. Each EEG record includes recordings from all electrodes, which is a multi-channel EEG signal, and hence has two dimensions of time and space. We often analyse each single-channel EEG signal in the joint time-frequency domain, thus adding an extra dimension of frequency. In special situations, there could be even 7 dimensions: time, frequency, space, trial, condition, subject and group [10]. Tensor decomposition reveals interactions among multiple dimensions, improving the quality and interpretation of the analysis. Other reasons for using tensor decomposition is to exploit its uniqueness, versatile representation and superior performance [12].

Incomplete observation of EEG signals can occur as well, when for example electrodes become loose or disconnected during the recording process. This is due to difficulty of keeping the head fixed (e.g., EEG recording for children) or reduced quality of conductive gels when the recording is done in a long time (e.g., 24-hour monitoring). In such cases, signals recorded from one or several electrodes do not correctly describe the electrical activity of the brain and thus can be discarded, making the observed data incomplete.

Most existing methods for EEG analysis by tensor decomposition are based on batch processing [10], [13] (i.e., data are stored and processed offline). However, when data are of streaming nature like EEG signals in long recordings, adaptive processing is more suitable. This is due to the fact that processing such kinds of data often requires fast updating instead of recalculating from the beginning or processing the whole data as batch method, because of time and storage constraints. To the best of our knowledge, tensor tracking from streaming EEG data has only been considered in [14]. However, the situation of incomplete data was not taken into account.

In this paper, we aim to improve on existing tensor tracking algorithms from incomplete tensors and to apply such an improvement to multi-channel EEG analysis. While there are different models of tensor decomposition, we focus here Parallel Factor (PARAFAC) decomposition. This is inspired by our two recent works. The first one is on adaptive PARAFAC tracking [6], which combines the Parallel Subspace Estimation and Tracking by Recursive Least Squares (PETRELS) algorithm proposed by Chi *et al.* [15] for subspace tracking and the adaptive PARAFAC decomposition algorithm proposed by Nion and Sidiropoulos [3] for streaming third-order tensors. The second one is on a new formulation of PETRELS cost function, which we will provide details in a subsequent publication for subspace tracking from incomplete data.

The contributions are three-fold. First, we propose a nonlinear formulation of the PETRELS cost function. The resulting nonlinear subspace tracking algorithm, referred to as NL-PETRELS, can converge faster than PETRELS while achieving a similar performance. Second, by replacing the subspace tracking step in our adaptive PARAFAC decomposition [6] with NL-PETRELS, we propose a non-linear tensor tracking algorithm for incomplete data. Third, we show how our tensor tracking algorithm can be used to track incomplete multi-channel EEG data.

Notations: Calligraphic letters are used for tensors. Boldface uppercase, boldface lowercase, and lowercase denote matrices, (row and column) vectors, and scalars respectively. Operators $\otimes$, $\odot$, $*$, $\circ$, $(\cdot)^T$ and $(\cdot)^{\#}$ denote the Kronecker product, the Khatri-Rao product, the Hadamard product (element-wise matrix product), and the outer product, the transpose and the pseudo-inverse, respectively.

II. Proposed Algorithms for Incomplete Data

A. Non-linear subspace tracking from incomplete data

Consider the standard linear data model [15] of $\mathbf{r}(t) \in \mathbb{R}^n$, given by

$$\mathbf{r}(t) = \mathbf{D}\mathbf{s}(t) + \mathbf{n}(t), \tag{1}$$

where $\mathbf{D} \in \mathbb{R}^{n\times p}$ is the system matrix of full column rank, $\mathbf{s}(t) \in \mathbb{R}^p$ is the signal vector randomly distributed according to the Gaussian distribution with zero mean and unit variance, and $\mathbf{n}(t) \in \mathbb{R}^n$ is the noise vector distributed according to the Gaussian distribution with zero mean and variance σ^2.

A partial observation of $\mathbf{r}(t)$ is given by

$$\mathbf{y}(t) = \mathbf{p}(t) * \mathbf{r}(t), \tag{2}$$

where $\mathbf{p}(t) = [p_1(t), p_2(t), \ldots, p_n(t)]^T$ is the mask vector; that is, $p_i(t) = 1$ if the i-th entry of $\mathbf{r}(t)$ is observed, and $p_i(t) = 0$ otherwise.

Our purpose is to estimate a principal subspace $\mathbf{W}$ of $\mathbf{D}$, given that the data were incompletely acquired according to (2). To do so, we first propose the following general non-linear cost function for subspace tracking in the situation of incomplete data:

$$J(\mathbf{W}) = \sum_{i=t-L+1}^{t} \beta^{t-i} \|\mathbf{P}(i)[\mathbf{y}(i) - \mathbf{W} g\left((\mathbf{P}(i)\mathbf{W})^{\#}\mathbf{y}(i)\right)]\|^2, \tag{3}$$

where L is the length of a window applied to the signal, β is known as the forgetting factor with $0 < \beta \leqslant 1$, $\mathbf{P}(t) = \mathrm{diag}(\mathbf{p}(t))$, and $g(x)$ is a non-linear function.

We have the following observations:

- If $g(x) = x$, we obtain a linear cost function. Specifically, the cost function in (3) corresponds to the exponential-window cost function when $L \to \infty$, and to the sliding-window cost function when $\beta = 1$. Moreover, for complete data (i.e., $\mathbf{P}(i) = \mathbf{I}$ for all i), (3) becomes the well-known projection approximation subspace tracking (PAST) cost function [16].
- In general, $g(x)$ can be any non-linear function whose specific form depends on the application at hand. For example, in this paper, we use $g(x) = \tanh(x)$ for subspace and tensor tracking, aimed at accelerating the convergence rate. We also note that (3) is essentially compatible with non-linear principal component analysis (PCA) investigated in [17], [18] for complete data.

In this paper, we present the proposed NL-PETRELS subspace tracking algorithm, only for the case of exponential-window cost function. Accordingly, (3) is rewritten as

$$J_{\mathrm{EW}}(\mathbf{W}) = \sum_{i=1}^{t} \beta^{t-i} \|\mathbf{P}(i)[\mathbf{y}(i) - \mathbf{W} g\left((\mathbf{P}(i)\mathbf{W})^{\#}\mathbf{y}(i)\right)]\|^2. \tag{4}$$

Following the derivation from [15] and [17], the proposed algorithm can be summarised as in Algorithm 1.

The main difference, compared to PETRELS, comes from the non-linear step at line 3 in estimating $\mathbf{a}(t)$ under the condition that the number of non-zero percentage (NNZP) is less than a certain threshold (ϵ_0), which is always relative small and determined by the experiment. For example, it will be set to be less than 10% in total observation in our simulation. Otherwise, the algorithm essentially corresponds to PETRELS.

B. Non-linear PARAFAC tracking from incomplete tensors

In this section, we generalize NL-PETRELS for adaptive tensor tracking of third-order tensors, following the PARAFAC decomposition model. A third-order tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ can be decomposed according to the PARAFAC model as [1]

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{5}$$

Algorithm 1: Nonlinear PETRELS (NL-PETRELS)

Initialization: Random $\mathbf{W}(0) \in \mathbb{R}^{n\times p}$, $\mathbf{R}_m^{-1}(0) = \mathbf{I}_p$

1 **for** $t = 1 : T$ **do**
2 **if** $\mathrm{NNZP} \leqslant \epsilon_0$ **then**
3 $\mathbf{a}(t) = g\left((\mathbf{P}(t)\mathbf{W}(t-1))^{\#}\mathbf{y}(t)\right)$
4 **end**
5 **else**
6 $\mathbf{a}(t) = (\mathbf{P}(t)\mathbf{W}(t-1))^{\#}\mathbf{y}(t)$
7 **end**
8 **end**
9 **for** $m = 1 : n$ **do**
10 $\alpha_m(t) = 1 + \beta^{-1}\mathbf{a}^T(t)\mathbf{R}_m^{-1}(t-1)\mathbf{a}(t)$
11 $\mathbf{u}_m(t) = \beta^{-1}\mathbf{R}_m^{-1}(t-1)\mathbf{a}(t)$
12 $\mathbf{R}_m^{-1}(t) = \beta^{-1}\mathbf{R}_m^{-1}(t-1) - p_m(t)\alpha_m^{-1}(t)\mathbf{u}_m(t)\mathbf{u}_m^T(t)$
13 $\mathbf{w}_m(t) = \mathbf{w}_m(t-1) + [y_m(t) - p_m(t)\mathbf{a}(t)\mathbf{w}_m(t-1)]\mathbf{R}_m^{-1}(t)\mathbf{a}(t)$
14 **end**

which is sum of R rank-one tensors[1]. Always, (5) is only an approximate tensor in a noisy environment, that is,

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r + \mathcal{N}, \tag{6}$$

where $\mathcal{N}$ is a noise tensor. By grouping $\mathbf{A} = [\mathbf{a}_1 \dots \mathbf{a}_R] \in \mathbb{R}^{I \times R}$, $\mathbf{B} = [\mathbf{b}_1 \dots \mathbf{b}_R] \in \mathbb{R}^{J \times R}$, and $\mathbf{C} = [\mathbf{c}_1 \dots \mathbf{c}_R] \in \mathbb{R}^{K \times R}$, (6) can be rewritten in matrix form[2] as

$$\mathbf{X} = (\mathbf{A} \odot \mathbf{C})\,\mathbf{B}^T + \mathbf{N}. \tag{7}$$

Thus, given a noisy data tensor $\mathcal{X}$, PARAFAC decomposition tries to perform R-rank best approximation in the least squares sense, that is,

$$\phi(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \| \mathbf{X} - (\mathbf{A} \odot \mathbf{C})\,\mathbf{B}^T \|_F \tag{8}$$

When the data are incomplete, (8) becomes

$$\phi_{\mathbf{M}}(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \| \mathbf{M} * \left(\mathbf{X} - (\mathbf{A} \odot \mathbf{C})\,\mathbf{B}^T\right) \|_F^2, \tag{9}$$

where $\mathbf{M}$ is a mask matrix, defined as

$$\mathbf{M}(i,j) = \begin{cases} 1, & \text{if } \mathbf{X}(i,j) \text{ was observed,} \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

In batch processing, the three dimensions of the tensor are constants. In adaptive processing, we are interested in this paper third-order tensors which have one dimension growing in time while the other two dimensions remain constant, e.g., $\mathcal{X}(t) \in \mathbb{R}^{I \times J(t) \times K}$, as shown at the top of Fig. 1.

Using the matrix representation in (7) and in the noiseless case, we have the following PARAFAC decompositions at two successive time instants $t-1$ and t:

$$\mathbf{X}(t-1) = [\mathbf{A}(t-1) \odot \mathbf{C}(t-1)]\,\mathbf{B}^T(t-1) \tag{11a}$$

$$\mathbf{X}(t) = [\mathbf{A}(t) \odot \mathbf{C}(t)]\,\mathbf{B}^T(t). \tag{11b}$$

Thus,

$$\mathbf{X}(t) = [\mathbf{X}(t-1) \quad \mathbf{x}(t)], \tag{12}$$

where $\mathbf{x}(t)$ is the vectorised representation of a new slice (see the bottom of Fig. 1):

$$\mathbf{x}(t) = [\mathbf{A}(t) \odot \mathbf{C}(t)]\,\mathbf{b}^T(t) = \mathbf{H}(t)\mathbf{b}^T(t), \tag{13}$$

where $\mathbf{b}^T(t)$ is the t-th column of $\mathbf{B}^T(t)$.

Consider the following exponentially weighted least-square cost function:

$$\Psi_{\mathbf{P}(t)}(t) = \sum_{i=1}^{t} \beta^{t-i} \| \mathbf{P}(i)[\mathbf{x}(i) - \mathbf{H}(t)\mathbf{b}^T(i)] \|^2. \tag{14}$$

Estimating the loading matrices of the adaptive PARAFAC model of (18) corresponds to

$$\underset{\mathbf{H}(t),\mathbf{B}(t)}{\text{minimize}}\ \Psi_{\mathbf{P}(t)}(t) \tag{15}$$

$$\text{subject to } \mathbf{H}(t) = \mathbf{A}(t) \odot \mathbf{C}(t). \tag{16}$$

[1]A rank-one tensor is defined as $\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$.

[2]Other matrix forms are possible.

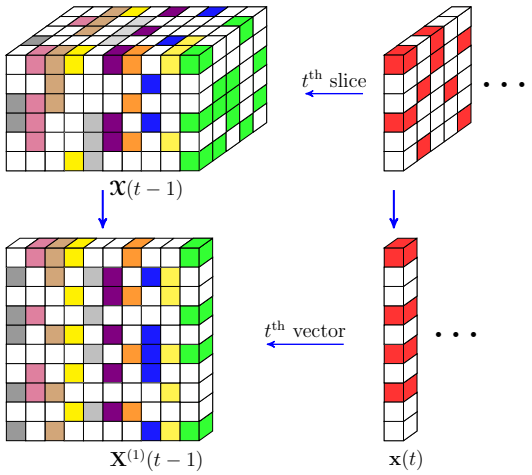


Fig. 1. Adaptive third-order tensor model for incomplete data and its equivalent matrix form.

We also adopt the following assumptions from [6]:

- The loading matrices $\mathbf{A}$ and $\mathbf{C}$ are unknown but follow slowly time-varying models, i.e., $\mathbf{A}(t) \simeq \mathbf{A}(t-1)$ and $\mathbf{C}(t) \simeq \mathbf{C}(t-1)$. As a consequence, since $\mathbf{H}(t) \simeq \mathbf{H}(t-1)$, we obtain

$$\mathbf{B}^T(t) \simeq \left[\mathbf{B}^T(t-1), \mathbf{b}^T(t)\right], \tag{17}$$

  which allows us to estimate $\mathbf{B}(t)$ in a simple manner. Specifically, instead of updating the whole $\mathbf{B}(t)$ at each time instant, we only need to estimate the *row* vector $\mathbf{b}(t)$ and augment it to $\mathbf{B}(t-1)$ to obtain $\mathbf{B}(t)$. In the other words, $\mathbf{B}(t)$ has time-shift structure.
- The tensor rank, R, is constant and known in advanced. Moreover, the uniqueness property of the new tensor is satisfied when a new data slice is added to the old tensor.

In the situation of incomplete data, $\mathbf{x}(t)$ is replaced by

$$\tilde{\mathbf{x}}(t) = \mathbf{p}(t) * \mathbf{x}(t), \tag{18}$$

where $\mathbf{p}(t)$ is defined in (2).

Observe that given $\mathbf{b}^T(t)$, estimating $\mathbf{H}(t)$ from incomplete observation $\tilde{\mathbf{x}}(t)$ is a least-squares problem. This procedure is known as alternating least-squares (ALS) minimization which is used extensively in the tensor literature. We also use this approach to develop our tensor tracking algorithm, which is summarised in Algorithm 2.

Given $\mathbf{H}(t-1)$, we can estimate $\mathbf{H}(t)$ by first setting

$$\mathbf{b}^T = g\left((\mathbf{P}(t)\mathbf{H}(t-1))^{\#}\tilde{\mathbf{x}}(t)\right), \tag{19}$$

at line 3 in Algorithm 1 of our proposed NL-PETRELS algorithm, then obtaining $\mathbf{H}(t)$ as the output of the algorithm.

To extract $\mathbf{A}(t)$ and $\mathbf{C}(t)$ from $\mathbf{H}(t)$, we use the bi-SVD method as in [6]:

$$\mathbf{a}_i(t) = \mathbf{H}_i^T(t)\mathbf{c}_i(t-1), \tag{20}$$

$$\mathbf{c}_i(t) = \frac{\mathbf{H}_i(t)\mathbf{a}_i(t)}{\| \mathbf{H}_i(t)\mathbf{a}_i(t) \|}, \tag{21}$$

with $i = 1, \ldots, R$. Note that each column of $\mathbf{H}(t)$ is the result of vectorising rank-1 matrix: $\mathbf{H}_i(t) = \text{unvec}(\mathbf{a}_i(t) \otimes \mathbf{c}_i(t))$. Thus, estimating vectors $\mathbf{c}_i(t)$ and $\mathbf{a}_i(t)$ corresponds to extract the principal left singular vector and the conjugate of the principal right singular vector of matrix $\mathbf{H}_i(t)$.

Finally, we re-estimate $\mathbf{b}^T(t)$ as

$$\mathbf{b}^T(t) = [\mathbf{P}(t)\,(\mathbf{A}(t) \odot \mathbf{C}(t))]^{\#}\,\tilde{\mathbf{x}}(t). \tag{22}$$

We note that when NNZP is small, computing $[\mathbf{P}(t)\,(\mathbf{A}(t) \odot \mathbf{C}(t))]^{\#}$ is fast because only non-zero rows of $\mathbf{H}(t)$ are used in the computation.

III. Experiments

In this section, we present selected experiments to illustrate the effectiveness of proposed algorithms. First, we assess tracking performance of the NL-PETRELS subspace tracking algorithm, using simulated data. Then, we illustrate how the NL-PETRELS-based PARAFAC tracking algorithm can be applied to real EEG data [19].

A. NL-PETRELS subspace tracking

To assess the accuracy of subspace estimation, we use (2) to generate simulated data and the following least-squares performance index [20]:

$$\text{SEP}(t) = \frac{\text{tr}\{\mathbf{W}_i^H(t)[\mathbf{I} - \mathbf{W}_{\text{ex}}(t)\mathbf{W}_{\text{ex}}^H(t)]\mathbf{W}_i(t)\}}{\text{tr}\{\mathbf{W}_i^H(t)(\mathbf{W}_{\text{ex}}(t)\mathbf{W}_{\text{ex}}^H(t))\mathbf{W}_i(t)\}}, \tag{23}$$

where $\mathbf{W}_i$ is the estimated subspace at the i-th run, and $\mathbf{W}_{\text{ex}}$ is the exact subspace weight matrix computed by orthorgonalising $\mathbf{A}$. The result is shown in Fig. 2.

We also assess performance through matrix completion example [15], as shown in Fig. 3. The MATLAB implementation of this experiment is downloaded from the web page of the first author. To assess convergence rate, we modify the codes to generate a sudden change of subspace at time instant $10,000$. Moreover, a noise level at 10^{-3} is added. In this experiment, normalized subspace error is used as performance index. For more details, we refer the reader to [15].

Parameters in both experiments are summarised in Table I. NNZP $= 0.1$ corresponds to only 10% observation data used. We used default parameters of PETRELS to have fair comparison in both experiments.

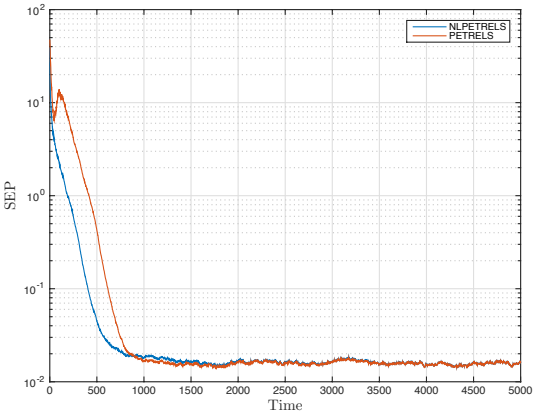


Fig. 2. NL-PETRELS subspace tracking performance.

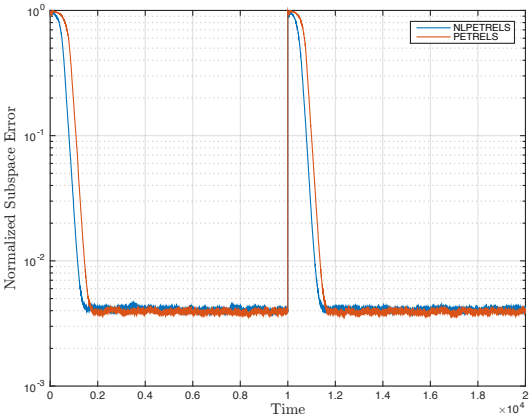


Fig. 3. Adaptive subspace tracking performance.

TABLE I
Experimental Parameters

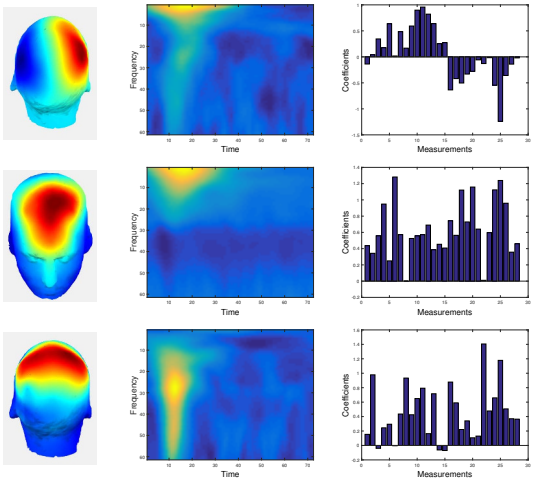
n	p	T	NNZP
500	10	5000/20000	0.1

Algorithm 2: NL-PETRELS-based PARAFAC tracking

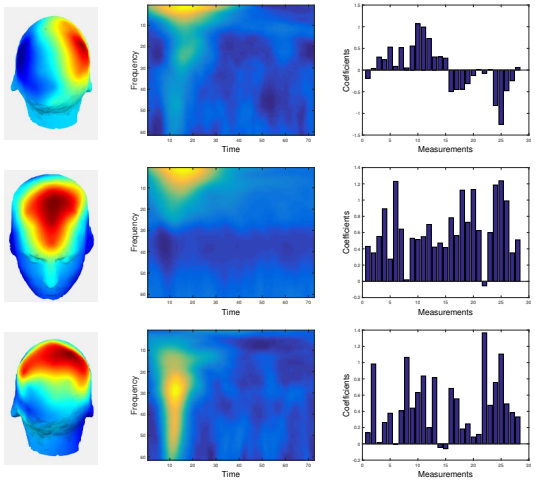
Initialization: $\mathbf{H}(0)$, $\mathbf{R}_m^{-1}(0) = \mathbf{I}_R$, $\mathbf{A}(0)$, $\mathbf{B}(0)$, $\mathbf{C}(0)$
for $t = 1 : T$ **do**
  $[\mathbf{H}(t), \mathbf{R}_m^{-1}(t), \mathbf{b}^T(t)] =$ NL-PETRELS $\left(\tilde{\mathbf{x}}(t), \mathbf{H}(t-1), \mathbf{R}_m^{-1}(t-1)\right)$
  for $i = 1 : R$ **do**
    $\mathbf{a}_i(t) = \mathbf{H}_i^T(t)\mathbf{c}_i(t-1)$
    $\mathbf{c}_i(t) = \dfrac{\mathbf{H}_i(t)\mathbf{a}_i(t)}{\| \mathbf{H}_i(t)\mathbf{a}_i(t) \|}$
  end
  $\mathbf{b}^T(t) = [\mathbf{P}(t)\,(\mathbf{A}(t) \odot \mathbf{C}(t))]^{\#}\,\tilde{\mathbf{x}}(t)$
end

We can see that in both experiments, when PETRELS and NL-PETRELS converge, they have the same performance. However, NL-PETRELS outperformed PETRELS in terms of convergence rate (first $1,000$ samples in the first experiment, and $2,000$ samples in the second one) and in presence of sudden change of subspace.

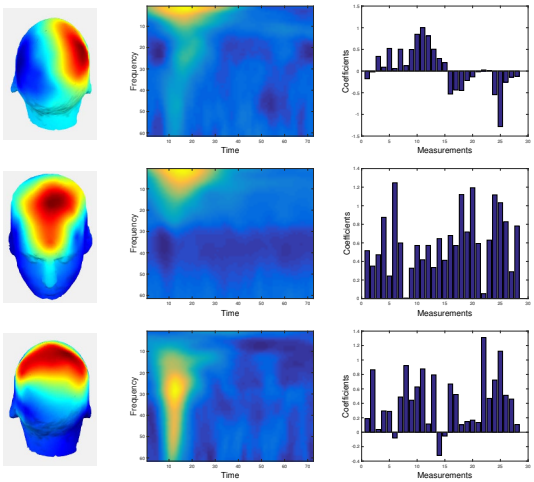
For non-linear characterization of the NL-PETRELS subspace tracking algorithm, as discussed in [18, Chapter 12], minimizing the non-linear cost function in (4) does not provide a smaller least mean square error than its linear version. This characterization also keeps in the situation of incomplete data and was confirmed by our experiments.



(a) CP-OPT for full data



(b) CP-WOPT for incomplete data



(c) NL-PETRELS based PARAFAC tracking for incomplete data

Fig. 4. Estimates of loading matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ using CP-WOPT and our proposed NL-PETRELS PARAFAC tracking.

B. NL-PETRELS based PARAFAC tracking from incomplete EEG data

We use the EEG dataset provided in [19], which records gamma activation during proprioceptive stimuli of left and right hands. The dataset includes 28 measurements of 14 subjects. For each subject, left and right hands are stimulated and recorded by 64 EEG channels.

The EEG data are represented by a tensor of three dimensions: *channel* $\times$ *time-frequency* $\times$ *measurement*. To create the time-frequency image from the EEG signal in each channel, the continuous wavelet transform was used [19]. This time-frequency matrix is then vectorised to form a vector of length 4392. Therefore, the size of the tensor is: $64 \times 4392 \times 28$.

We compare our NL-PETRELS-based PARAFAC tracking algorithm with the CP-WOPT algorithm in [21]. CP-WOPT is a batch algorithm for incomplete data. Accordingly, we process the data in a similar manner. The tensor is centered (demeaned) across the channels. The rank of the tensor is $R = 3$. To create the situation of incomplete data, for each measurement, data from randomly selected 20 channels are discarded. Different from CP-WOPT is the ability to deal with streaming data of our proposed algorithm. The implementation of this experiment used several MATLAB toolboxes: Tensor [22], Poblano [23], and EEGLAB [24].

The adaptivity is done along the second dimension (time-frequency), as if each EEG time-frequency image is vectorised and the resulting vector of data is being streamed. To initialize our algorithm, we run CP-WOPT with the first 1500 slices, i.e., tensor with size of $64 \times 1500 \times 28$. This is known as batch initialization [3] and is necessary to make algorithm converge. We have experimentally observed that random initialization may cause algorithm diverge for the EEG data.

The results are given in Fig. 4. Three rows in each sub-figure correspond to three PARAFAC components ($R = 3$), i.e. the first, second and third columns of the loading matrices. In each row, the 3-dimensional head, the time-frequency representation

and the bar plot correspond to the i-th vectors of the loading matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ respectively, $i = 1, 2, 3$. Fig. 4 illutrates the estimation of the loading matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, using CP-WOPT in (a) for full data and (b) for incomplete data and (c) using our proposed NL-PETRELS PARAFAC tracking for incomplete data, showing that our algorithm can track the loading matrices successfully.

In our experiment, for illustration purposes, the way we created the EEG tensor is offline, that is applying the continuous wavelet transform for the whole duration of the EEG signal in each channel and performed the tracking as if we gradually received data from this whole time-frequency vector. In practice, it would be more appropriate to perform the wavelet transform in real-time [25]–[28], as the time samples of an EEG signal is being recorded.

IV. Conclusion

In the context of using tensor decomposition in challenging situations where the observed data are streaming and incomplete, we have proposed a non-linear formulation of the PETRELS cost function and based on which we proposed NL-PETRELS subspace and tensor tracking algorithms. While the performance of the NL-PETRELS subspace tracking algorithm was investigated and shown to be better than PETRELS in terms of convergence rate, the NL-PETRELS based PARAFAC tracking algorithm was illustrated for tracking multi-channel incomplete EEG data, represented by a tensor of three dimensions: channel $\times$ vectorised time-frequency $\times$ measurement. The algorithm successfully tracked the data even when data from 20 out ouf 64 channels were missing. Investigation on the performance of the proposed tensor tracking algorithm by itself and with respect to the presented type of EEG tensor is necessary, as well as on different types of EEG tensors.

Acknowledgment

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.02-2015.32.

References

[1] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[2] A. Cichocki, D. Mandic, L. De 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 Processing Magazine*, vol. 32, no. 2, pp. 145–163, 2015.

[3] D. Nion and N. D. Sidiropoulos, "Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2299–2310, 2009.

[4] V.-D. Nguyen, K. Abed-Meraim, and N. Linh-Trung, "Second-order optimization based adaptive PARAFAC decomposition of three-way tensors," *Digital Signal Processing*, vol. 63, pp. 100–111, Apr. 2017.

[5] M. Mardani, G. Mateos, and G. B. Giannakis, "Subspace learning and imputation for streaming big data matrices and tensors," *IEEE Transactions on Signal Processing*, vol. 63, no. 10, pp. 2663–2677, 2015.

[6] T. Minh-Chinh, V.-D. Nguyen, N. Linh-Trung, and K. Abed-Meraim, "Adaptive PARAFAC decomposition for third-order tensor completion," in *6th IEEE International Conference on Communications and Electronics (ICCE)*, Jul. 2016, pp. 297–301.

[7] H. Kasai, "Online low-rank tensor subspace tracking from incomplete data by CP decomposition using recursive least squares," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 2519–2523.

[8] L. Sörnmo and P. Laguna, *Bioelectrical Signal Processing in Cardiac and Neurological Applications*. Academic Press, 2005.

[9] 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.

[10] F. Cong, Q.-H. Lin, L.-D. Kuang, X.-F. Gong, P. Astikainen, and T. Ristaniemi, "Tensor decomposition of EEG signals: A brief review," *Journal of Neuroscience Methods*, vol. 248, pp. 59–69, 2015.

[11] B. Hunyadi, P. Dupont, W. Van Paesschen, and S. Van Huffel, "Tensor decompositions and data fusion in epileptic electroencephalography and functional magnetic resonance imaging data," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 1, 2017.

[12] A. Cichocki, "Tensors decompositions: New concepts for brain data analysis?" *Journal of Control Measurement, and System Integration*, vol. 7, pp. 507–517, 2011.

[13] V. D. Nguyen, K. Abed-Meraim, and N. Linh-Trung, "Fast tensor decompositions for big data processing," in *International Conference on Advanced Technologies for Communications*, Oct. 2016, pp. 215–221.

[14] A. Fotouhi, E. Eqlimi, and B. Makkiabadi, "Evaluation of adaptive PARAFAC alogorithms for tracking of simulated moving brain sources," in *37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Aug. 2015, pp. 3819–3822.

[15] Y. Chi, Y. C. Eldar, and R. Calderbank, "PETRELS: Parallel subspace estimation and tracking by recursive least squares from partial observations," *IEEE Transactions on Signal Processing*, vol. 61, no. 23, pp. 5947–5959, Dec. 2013.

[16] B. Yang, "Projection approximation subspace tracking," *IEEE Transactions on Signal Processing*, vol. 43, no. 1, pp. 95–107, 1995.

[17] J. Karhunen and J. Joutsensalo, "Representation and separation of signals using nonlinear PCA type learning," *Neural Networks*, vol. 7, no. 1, pp. 113–127, 1994.

[18] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. John Wiley & Sons, 2004, vol. 46.

[19] M. Mørup, L. K. Hansen, and S. M. Arnfred, "ERPWAVELAB: A toolbox for multi-channel analysis of timefrequency transformed event related potentials," *Journal of Neuroscience Methods*, vol. 161, no. 2, pp. 361–368, 2007.

[20] V.-D. Nguyen, K. Abed-Meraim, N. Linh-Trung, and R. Weber, "Generalized minimum noise subspace for array processing," *IEEE Transactions on Signal Processing*, vol. 65, no. 14, pp. 3789–3802, Jul. 2017.

[21] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup, "Scalable tensor factorizations for incomplete data," *Chemometrics and Intelligent Laboratory Systems*, vol. 106, no. 1, pp. 41–56, 2011, multiway and Multiset Data Analysis.

[22] B. W. Bader, T. G. Kolda *et al.*, "MATLAB tensor toolbox version 2.6," Feb. 2015. [Online]. Available: http://www.sandia.gov/~tgkolda/TensorToolbox/

[23] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Poblano v1.0: A MATLAB toolbox for gradient-based optimization," Sandia National Laboratories, Tech. Rep. SAND2010-1422, 2010.

[24] A. Delorme and S. Makeig, "EEGLAB: An open source toolbox for analysis of single-trial EEG dynamics including independent component analysis," *J. Neuroscience Methods*, vol. 134, no. 1, pp. 9–21, 2004.

[25] K. McGill and C. Taswell, "Length-preserving wavelet transform algorithms for zero-padded and linearly-extended signals," *preprint, Rehabilitation R&D Center, VA Medical Center, Palo Alto, CA*, 1992.

[26] M. Vishwanath, "The recursive pyramid algorithm for the discrete wavelet transform," *IEEE Transactions on Signal Processing*, vol. 42, no. 3, pp. 673–676, 1994.

[27] H. O. Mota, F. H. Vasconcelos, and R. M. da Silva, "Real-time wavelet transform algorithms for the processing of continuous streams of data," in *IEEE International Workshop on Intelligent Signal Processing*. IEEE, 2005, pp. 346–351.

[28] P. Rajmic and J. Vlach, "Real-time audio processing via segmented wavelet transform," in *International Conference on Digital Audio Effects (DAFx-07)*, 2007.