Accurate PARAFAC2 Decomposition for Temporal Irregular Tensors with Missing Values

Jun-Gi Jang
Seoul National University
Seoul, Republic of Korea
elnino4@snu.ac.kr

Jeongyoung Lee
Seoul National University
Seoul, Republic of Korea
ljklee@snu.ac.kr

Jiwon Park
Seoul National University
Seoul, Republic of Korea
jiwon_p@snu.ac.kr

U Kang
Seoul National University
Seoul, Republic of Korea
ukang@snu.ac.kr

***Abstract*—Given a temporal irregular tensor with missing values, how can we perform accurate decomposition for the tensor? Many real-world data can be represented as a temporal irregular tensor which is a collection of matrices whose rows corresponding to the time dimension have different sizes, but columns have the same size. PARAFAC2 decomposition is a powerful tool for analyzing an irregular tensor in many interesting applications such as phenotype discovery and fault detection. However, existing PARAFAC2 decomposition methods fail to handle irregular tensors with missing values since they treat the missing values as zeros. Furthermore, few methods that utilize temporal regularization focus only on a specific type of temporal irregular tensors.**

In this paper, we propose ATOM, an accurate PARAFAC2 decomposition method which carefully handles missing values in a temporal irregular tensor. ATOM provides a reformulated loss function that fully excludes missing values and accurately updates factor matrices by considering sparsity patterns of each row. ATOM also captures temporal patterns by exploiting smoothing regularization with time dependency. Extensive experiments show that ATOM provides up to $7.9\times$ lower error rate than existing PARAFAC2 decomposition methods.

***Index Terms*—irregular tensor, parafac2 decomposition, missing value**

I. Introduction

Given a temporal irregular tensor with missing values, how can we accurately find latent factors by decomposition? A temporal irregular tensor is a natural representation of real-world data as a collection of matrices whose rows corresponding to the time dimension have different sizes, but columns have the same size. For example, stock data can be represented as a temporal irregular tensor where all the stocks have the same number of features but their listing periods are different. Traffic data and movie rating data can be also represented as temporal irregular tensors.

Many research works has studied tensor decomposition for improving its performance [1]–[7] and applying it to real-world applications including concept discovery [8], [9], forecasting [10], [11], anomaly detection [12]–[14], and knowledge base completion [15]. It extracts meaningful information from tensors by learning latent factor matrices. Among many tensor decomposition methods, PARAFAC2 decomposition is tailored for analyzing an irregular tensor by approximating it into latent factor matrices. Several applications using PARAFAC2 decomposition include phenotype discovery [16]–[18], trend analysis [19], and heart failure prediction [20].

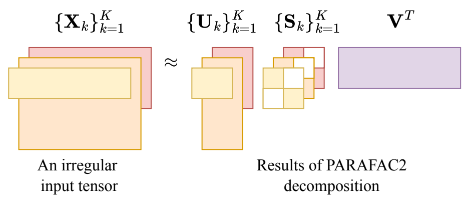

$\{\mathbf{X}_k\}_{k=1}^K$
$\approx$
$\{\mathbf{U}_k\}_{k=1}^K$
$\{\mathbf{S}_k\}_{k=1}^K$
$\mathbf{V}^T$
An irregular input tensor
Results of PARAFAC2 decomposition

Fig. 1. PARAFAC2 decomposition converts each slice matrix $\mathbf{X}_k$ into $\mathbf{U}_k \in \mathbb{R}^{I_k \times R}$, $\mathbf{S}_k \in \mathbb{R}^{R \times R}$, and $\mathbf{V} \in \mathbb{R}^{J \times R}$ for $k = 1, ..., K$, where R is a target rank and $\mathbf{S}_k$ is a diagonal matrix.

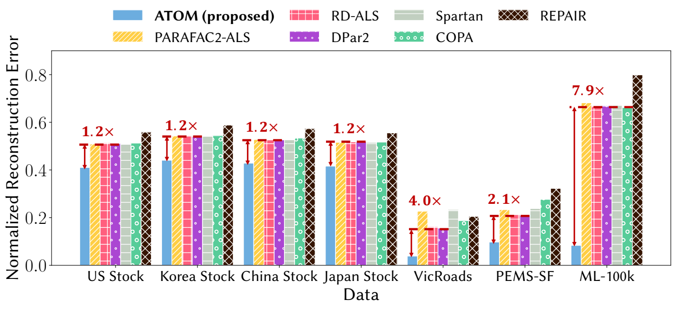


Fig. 2. Evaluation of missing value prediction for test ratio 30%. ATOM outperforms existing PARAFAC2 decomposition methods, providing up to $7.9\times$ lower error rate than the competitors.

Many recent works [14], [16]–[18], [21] have utilized PARAFAC2 decomposition (see Fig. 1) to analyze irregular tensors. They adopt alternating optimization (AO) which alternatively updates factor matrices of PARAFAC2 decomposition. However, they fail to obtain accurate factor matrices in temporal irregular tensors with missing values since they set the missing values to zeros. Although few methods [17], [18] utilize temporal regularization, they can be applied only in a tensor with a special structure (e.g., non-negativity). The main challenges to be tackled are 1) how to handle missing values, 2) how to capture temporal patterns, and 3) how to handle sparsity patterns in a given tensor.

In this paper, we propose ATOM, an Accurate PARAFAC2 decomposition method for Temporal irregular tensOrs with Missing values. We provide a reformulated loss function with a scalar form, which excludes missing values completely in the loss function. In addition, we add effective temporal regularization that makes nearby temporal factor vectors similar to each other. Finally, we carefully derive row-wise update rules to minimize the reformulated loss function. With these ideas, ATOM performs a highly accurate decomposition in

TABLE I
SYMBOL DESCRIPTION.

Symbol	Description
$\{\mathbf{X}_k\}_{k=1}^K$	irregular tensor of slices $\mathbf{X}_k$ for $k = 1, ..., K$
$\mathbf{X}_k$	slice matrix ($\in I_k \times J$)
$\mathbf{X}(i, :)$	i-th row vector of a matrix $\mathbf{X}$
$\mathbf{X}(:, j)$	j-th column vector of a matrix $\mathbf{X}$
$\mathbf{X}(i, j)$	(i, j)-th element of a matrix $\mathbf{X}$
$\mathbf{U}_k$, $\mathbf{S}_k$, $\mathbf{Q}_k$	factor matrices of the kth slice
$\mathbf{H}$, $\mathbf{V}$	factor matrices of an irregular tensor
R	target rank
$\odot$	Khatri-Rao product
$*$	element-wise product
$vec(\cdot)$	vectorization of a matrix

temporal irregular tensors with missing values, achieving up to 7.9× lower error rate than existing PARAFAC2 decomposition methods.

Our main contributions are summarized as follows:

- **Method.** we propose ATOM, an accurate PARAFAC2 decomposition method for temporal irregular tensors with missing values.
- **Theory.** We analyze the convergence of ATOM, and show that ATOM monotonically decreases the loss function.
- **Experiments.** Extensive experiments show that ATOM provides up to 7.9× lower error rate than existing PARAFAC2 decomposition methods (see Fig. 2).

The code and datasets are available at **https://datalab.snu.ac.kr/atom**.

II. PRELIMINARIES

In this section, we describe tensor notations, tensor operations, and PARAFAC2 decomposition. Table I presents the symbols used in this paper.

Boldface lowercases (e.g. $\mathbf{x}$) and boldface capitals (e.g. $\mathbf{X}$) denote vectors and matrices, respectively. Note that indices start at 1 in this paper.

Irregular Tensor. An irregular tensor is denoted by $\{\mathbf{X}_k\}_{k=1}^K$ consisting of a slice matrix $\mathbf{X}_k \in \mathbb{R}^{I_k \times J}$ where K is the number of slice matrices. Note that the column size J is the same for all the slice matrices while their row sizes I_k are different.

Frobenius Norm. The Frobenius norm of $\mathbf{X} \in \mathbb{R}^{I_k \times J}$ is denoted by $\|\mathbf{X}\|_F$ and defined as follows:

$$\|\mathbf{X}\|_F = \sqrt{\sum_{i,j} (\mathbf{X}(i,j))^2}.$$

where $\mathbf{X}(i, j)$ is the (i, j)-th element of $\mathbf{X}$.

Khatri-Rao product. Given two matrices $\mathbf{X} \in \mathbb{R}^{p \times q}$ and $\mathbf{Y} \in \mathbb{R}^{r \times q}$, the Khatri-Rao product is denoted by $(\mathbf{X} \odot \mathbf{Y}) \in \mathbb{R}^{pr \times q}$. The Khatri-Rao product performs the Kronecker product column by column: $(\mathbf{X} \odot \mathbf{Y}) = [\mathbf{X}(:,1) \otimes \mathbf{Y}(:,1) \,\|\, \cdots \,\|\, \mathbf{X}(:,q) \otimes \mathbf{Y}(:,q)]$, where $\otimes$ is Kronecker product and $\|$ denotes the horizontal concatenation. Each element of $\mathbf{a} = \mathbf{X}(:,q) \otimes \mathbf{Y}(:,q) \in \mathbb{R}^{pr}$ is defined as follows:

$$\mathbf{a}((i-1)r + k) = \mathbf{X}(i,q)\mathbf{Y}(k,q)$$

where $\mathbf{a}((i-1)r + k)$ is the $((i-1)r + k)$th element of the vector $\mathbf{a}$.

PARAFAC2 decomposition. Harshman [22] proposed PARAFAC2 decomposition which effectively analyzes irregular tensors. The definition of PARAFAC2 decomposition is as follows. As shown in Fig. 1, PARAFAC2 decomposition decomposes each k-th slice matrix $\mathbf{X}_k \in \mathbb{R}^{I_k \times J}$ into $\mathbf{U}_k\mathbf{S}_k\mathbf{V}^T$ when a target rank R and a 3-order irregular tensor $\{\mathbf{X}_k\}_{k=1}^K$ are given. Note that $\mathbf{U}_k$ is a matrix of the size $I_k \times R$, $\mathbf{S}_k$ is a diagonal matrix of the size $R \times R$, and $\mathbf{V}$ is a matrix of the size $J \times R$ which is common for all the slices.

The objective function of PARAFAC2 decomposition [22] is given as follows.

$$\min_{\{\mathbf{U}_k\},\{\mathbf{S}_k\},\mathbf{V}} \sum_{k=1}^{K} ||\mathbf{X}_k - \mathbf{U}_k\mathbf{S}_k\mathbf{V}^T||_F^2 \qquad (1)$$

Many PARAFAC2 decomposition methods [16]–[18], [21], [23] solve the above objective function (Eq. (1)) by adopting Alternating Optimization which iteratively updates a target factor matrix while fixing all the other factor matrices. Cheng et al. [21] and Jang et al. [23] focus on improving the efficiency of PARAFAC2 decomposition. Other works [16]–[18] focus on analyzing EHR data represented as a sparse irregular tensor with non-negative values by finding interpretable factor matrices. They use regularizations to improve interpretability. However, all the methods mentioned above fail to deal with an irregular tensor with missing values since they do not exclude missing values from the objective function (Eq. (1)). Furthermore, temporal regularization used in the previous works [17], [18] is limited since it is applicable only to a specific type of irregular tensors (e.g., EHR data).

III. PROPOSED METHOD

In this paper, we propose ATOM, an accurate temporal irregular tensor decomposition method handling missing values. We need to tackle the following challenges in obtaining accurate factor matrices of PARAFAC2 decomposition:

C1. **Fully excluding missing values in a loss function.** Previous works handle missing values as zeros. How can we avoid handling missing values in the loss function?

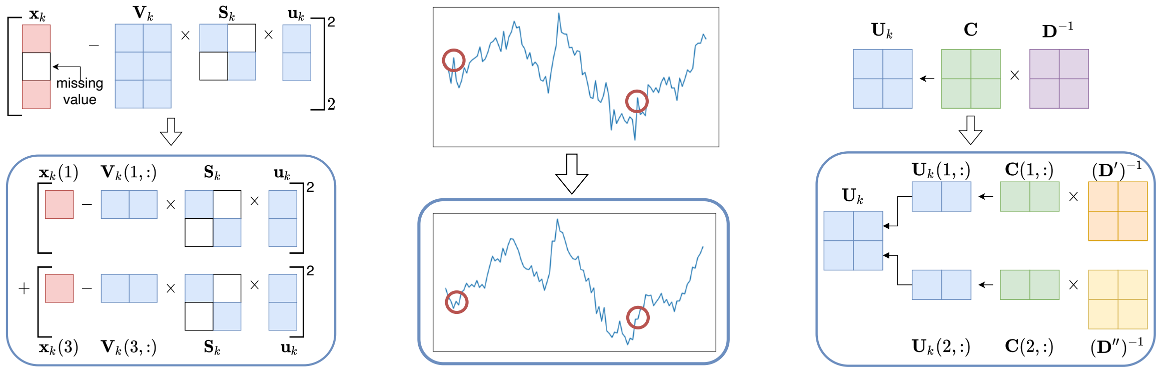
C2. **Capturing temporal patterns.** Temporal patterns are inherent in many temporal irregular tensors. How can we capture the temporal patterns in the tensor?

C3. **Optimizing an update procedure.** Previous update procedures are based on the existing loss function with the matrix form. However, they conflict with a loss function that excludes missing values. How can we design an update procedure for factor matrices of PARAFAC2 decomposition?

We address the above challenges with the following ideas (see Fig. 3):

I1. **Reformulating the loss function of PARAFAC2 decomposition as a scalar form (Section III-A).** A loss function with the scalar form fully exclude missing values when updating factor matrices.

I2. **Temporal regularization (Section III-B).** We regularize nearby temporal factor vectors to be similar to each other. Temporal factor vectors change smoothly over time.



(a) Loss function with a scalar form (b) Temporal Smoothness (c) Update row by row

Fig. 3. Description of what we modify. Blue boxes indicate examples for main ideas of ATOM. (a) We reformulate a loss function with a matrix form into that with a scalar form. It enables us to fully exclude missing values in an irregular tensor. (b) Temporal regularization makes the temporal factor matrix $\mathbf{U}_k$ smooth for the time dimension by making nearby factor vectors similar. As you can see in the red circles, Our temporal factor vector $\mathbf{U}_k(:,r)$ is smoother than the vector updated without temporal smoothness. (c) We update factor matrices row by row since rows of a factor matrix are updated by using different inverse terms. Rows of a slice matrix have different sparsity patterns which affect an update procedure derived from a reformulated loss function with a scalar form (lower one of Fig. 3(a)). A previous update rule derived from a loss function with a matrix form (upper one of Fig. 3(a)) does not need to consider sparsity patterns of missing values since all rows of a factor matrices use the same inverse term.

I3. **Row-wise update procedure for factor matrices (Section III-C).** We attain a highly accurate decomposition by a row-wise update procedure tailored for the reformulated loss function.

A. *Loss Function with Scalar Form*

Our objective is to fully exclude missing values from the loss function (Eq. (1)), and reformulate the loss function without them. We peer into the problem of the loss function (Eq. (1)) in terms of missing values, and then provide a solution to fully exclude them.

Problem. In the previous loss function (Eq. (1)), computing the value $||\mathbf{X}_k - \mathbf{U}_k\mathbf{S}_k\mathbf{V}^T||_F^2$ of the Frobenius norm requires setting all the missing values to 0, which leads to a degraded performance.

Example. We provide a toy example to illustrate the problem. Assume that we minimize a loss function, $\|\mathbf{x} - \mathbf{VSu}\|_2^2$, using alternating optimization where $\mathbf{x} \in \mathbb{R}^3$ is an input vector whose second element has a missing value, $\mathbf{u} \in \mathbb{R}^R$ is a factor vector, $\mathbf{S} \in \mathbb{R}^{R\times R}$ is a diagonal factor matrix, $\mathbf{V} \in \mathbb{R}^{3\times R}$ is a factor matrix, and R is a target rank. Then, we update $\mathbf{u}$ with the following rule while fixing $\mathbf{S}$ and $\mathbf{V}$:

$$\mathbf{u} \leftarrow \left(\mathbf{SV}^T\mathbf{VS}\right)^{-1}\mathbf{SV}^T\mathbf{x} \tag{2}$$

The above update rule is derived from $\frac{\partial L}{\partial \mathbf{u}_k} = 0$ where $L = \|\mathbf{x} - \mathbf{VSu}\|_2^2$. The problem of Eq. (2) is that $\mathbf{u}$ is updated to make $\mathbf{V}(2,:)\mathbf{Su}$ close to 0 where $\mathbf{V}(2,:) \in \mathbb{R}^{1\times R}$ is the second row vector of $\mathbf{V}$. This is because $\|\mathbf{x} - \mathbf{VSu}\|_2^2$ is minimized when $\mathbf{V}(2,:)\mathbf{Su}$ is equal to zero. However, this is not what we want since the missing value is not necessarily 0. Previous PARAFAC2 decomposition methods based on alternating optimization have slightly different update procedures, but the concept is implicit in their procedures.

Solution. How can we fully exclude missing values when updating factor matrices? As shown in Fig. 3(a), we reformulate the loss function for PARAFAC2 decomposition as a scalar form. Given an irregular tensor $\{\mathbf{X}_k\}_{k=1}^K$ with observable entries $\{\Omega_k\}_{k=1}^K$, we aim to find factor matrices, $\mathbf{U}_k$, $\mathbf{S}_k$, and $\mathbf{V}$, which approximates the given tensor by minimizing the following loss function with the scalar form:

$$\mathcal{L} = \sum_{k=1}^{K}\sum_{(i,j)\in\Omega_k}\left(\mathbf{X}_k(i,j) - \mathbf{U}_k(i,:)\mathbf{S}_k\mathbf{V}(j,:)^T\right)^2 \tag{3}$$

where $\mathbf{X}_k$ is the k-th slice of the given irregular tensor, and Ω_k is the observable entries of the kth slice matrix. The loss function (Eq. (3)) involves only observable entries Ω_k for all k. With the function, we accurately update factor matrices using only observable entries Ω_k for all k.

Example. We provide an example to compare our loss function with the previous function. We first reformulate $\|\mathbf{x} - \mathbf{VSu}\|_2^2$ into $(\mathbf{x}(1) - \mathbf{V}(1,:)\mathbf{Su})^2 + (\mathbf{x}(3) - \mathbf{V}(3,:)\mathbf{Su})^2$ where $\mathbf{x}(i)$ is the ith element of the vector $\mathbf{x}$. We adopt alternating optimization, and update $\mathbf{u}$ while fixing $\mathbf{S}$ and $\mathbf{V}$. Then, updating $\mathbf{u}$ is computed:

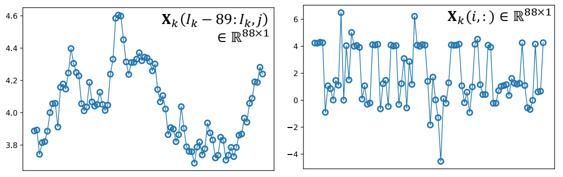
$$\begin{aligned}\mathbf{u} \leftarrow &\left(\mathbf{S}\left(\mathbf{V}(1,:)^T\mathbf{V}(1,:) + \mathbf{V}(3,:)^T\mathbf{V}(3,:)\right)\mathbf{S}\right)^{-1}\\ &\mathbf{S}\left(\mathbf{V}(1,:)^T\mathbf{x}(1) + \mathbf{V}(3,:)^T\mathbf{x}(3)\right)\end{aligned} \tag{4}$$

Note that $\mathbf{S}\left(\mathbf{V}(1,:)^T\mathbf{x}(1) + \mathbf{V}(3,:)^T\mathbf{x}(3)\right)$ is equal to $\mathbf{SV}^T\mathbf{x}$ when $\mathbf{x}(2)$ is set to 0 in $\mathbf{SV}^T\mathbf{x}$. The main difference between Eq. (4) and (2) is the inside of the inverse term. In contrast to Eq. (2), the term, $\mathbf{SV}(2,:)^T\mathbf{V}(2,:)\mathbf{S}$, is fully excluded in Eq. (4). This leads to accurate update of u by not interpreting missing values to zero. In summary, the reformulation with the scalar form is simple but powerful for excluding missing values completely and laying the groundwork for updating factor matrices accurately.

B. *Regularizations*

We add effective regularizations that mix well with the loss function (Eq. (3)).

Temporal Smoothness. We add temporal smoothness regularization such that the values of temporal factor matrices change gradually over time. Temporal regularization used in few PARAFAC2 decomposition methods [17], [18] is effective only on specific data with non-negativity. Therefore, we use more general and effective temporal smoothness applicable to various irregular tensors, capturing temporal patterns which



(a) a column vector of $\mathbf{X}_k$ (b) a row vector of $\mathbf{X}_k$

Fig. 4. An example of a real-world slice matrix in US Stock data. Rows and columns of $\mathbf{X}_k$ correspond to the time and feature dimensions, respectively. The left and right figures show opening prices (one of features) during 88 days, and values of 88 features at one day, respectively. Compared to the values in the right figure, the values gradually change in the left figure.

change gradually over time as described in Fig. 4. Our idea is to make nearby temporal factor vectors $\mathbf{U}_k(i,:)$ similar to each other. It helps capture temporal patterns and thus improves the performance of tensor decomposition [11], [24]. The main difference from the previous works [11], [20], [24] that handle regular tensors or a binary irregular tensor comes from the fact that ATOM focuses on analyzing real-valued irregular tensors with missing values and using PARAFAC2 decomposition. We add the following regularization in the loss function (Eq. (3)):

$$\lambda_s \sum_{i=1}^{I_k-1} \|\mathbf{U}_k(i,:) - \mathbf{U}_k(i+1,:)\|_2^2 \qquad (5)$$

where λ_s is a hyperparameter to adjust the effect of temporal smoothness. As shown in the example of Fig. 3(b), this regularization enforces temporal factor vectors to change gradually along the time axis.

Uniqueness. Recent works [16]–[18], [21], [23] use uniqueness regularization for the interpretability of factor matrices. They replace $\mathbf{U}_k$ with $\mathbf{Q}_k\mathbf{H}$ for uniqueness as proved in many works [25]–[27] where $\mathbf{Q}_k$ is a column orthogonal matrix. Then, they approximate $\mathbf{X}_k$ into $\mathbf{Q}_k\mathbf{H}\mathbf{S}_k\mathbf{V}^T$ which are unique factor matrices except for scaling and permutation of the factor vectors. Note that non-unique factor matrices make interpretation difficult since there can be several rotated factor matrices that cast doubt on the reliability of interpretation. Therefore, preserving uniqueness improves the interpretability of factor matrices. However, the explicit replacement provokes an accuracy issue due to under-fitting. Therefore, we add Eq. (6) to Eq. (3), instead of the explicit replacement.

$$\lambda_u \|\mathbf{U}_k - \mathbf{Q}_k\mathbf{H}\|_F^2 \qquad (6)$$

where λ_u is a hyperparameter to adjust the effect of uniqueness.

L2 regularization. We apply L2 regularization to $\mathbf{V}$ and $\mathbf{S}_k$.

$$\lambda_l \|\mathbf{V}\|_F^2 \quad \text{and} \quad \lambda_l \sum_{k=1}^{K} \|\mathbf{S}_k\|_F^2 \qquad (7)$$

where λ_l is a hyperparameter of L2 regularization. L2 regularization has been widely used for avoiding overfitting. Furthermore, this regularization prevents the computation of the inverse of a singular matrix in our update procedure. For example, let the column size of $\mathbf{V}$ be $R = 10$ in Eq. (4). Then, $\mathbf{S}_k\left(\mathbf{V}(1,:)^T\mathbf{V}(1,:) + \mathbf{V}(3,:)^T\mathbf{V}(3,:)\right)\mathbf{S}_k \in \mathbb{R}^{10\times10}$ is a singular matrix since its rank is equal to 2 and is smaller than 10. In other words, we fail to update $\mathbf{u}_k$ when the number of observed values in the row is smaller than the target rank R. Applying L2 regularization transforms the inside of the inverse term into $\lambda_l\mathbf{I} + \mathbf{S}_k\left(\mathbf{V}(1,:)^T\mathbf{V}(1,:) + \mathbf{V}(3,:)^T\mathbf{V}(3,:)\right)\mathbf{S}_k$. Therefore, we avoid computing the inverse of a singular matrix by using L2 regularization. The detail of the derivation is described in the next section.

Loss function. We propose the following loss function, which fully excludes missing values and includes effective regularizations:

$$\begin{aligned}\mathcal{L} = \sum_{k=1}^{K}\Bigg(&\sum_{(i,j)\in\Omega_k}\left(\mathbf{X}_k(i,j) - \mathbf{U}_k(i,:)\mathbf{S}_k\mathbf{V}(j,:)^T\right)^2\\ &+ \lambda_s\sum_{i=1}^{I_k-1}\|\mathbf{U}_k(i,:) - \mathbf{U}_k(i+1,:)\|_2^2\\ &+ \lambda_u\|\mathbf{U}_k - \mathbf{Q}_k\mathbf{H}\|_F^2 + \lambda_l\|\mathbf{S}_k\|_F^2\Bigg) + \lambda_l\|\mathbf{V}\|_F^2\end{aligned} \qquad (8)$$

C. Row-wise Update Procedure

Our goal is to update factor matrices that minimize the loss function (Eq. (8)). To achieve the goal, we need to consider sparsity patterns in an irregular tensor. Missing values in a tensor are not aligned but spread out all over the tensor. In addition, our loss function is in the scalar form, not the matrix form. However, existing AO (Alternating Optimization)-based methods, which alternatively update factor matrices, cannot handle the loss function (Eq. (8)) with the scalar form since they minimize the loss function with the matrix form by updating the entire elements of a factor matrix at once. In other words, they cannot address the problem of using different inverse terms for rows of a factor matrix, which is important to achieve high accuracy of decomposition. Therefore, we need to design a new update procedure appropriate for the reformulated function. Our idea is to update factor matrices row by row since rows of a slice matrix have different sparsity patterns and all the rows of a factor matrix are independent. Note that sparsity patterns of a slice matrix make rows of a factor matrix have different inverse terms. In the loss function (Eq.(3)), the inside of the square term includes the row vectors of $\mathbf{U}$, $\mathbf{W}$, and $\mathbf{V}$ where $\mathbf{W} \in \mathbb{R}^{K\times R}$ is a matrix whose kth row contains the diagonal elements of $\mathbf{S}_k$ (i.e., $\mathbf{W}(k,r) = \mathbf{S}_k(r,r)$). Therefore, we easily take the derivative with respect to a row of a factor matrix, $\frac{\partial\mathcal{L}}{\partial\mathbf{U}_k(i,:)}$, $\frac{\partial\mathcal{L}}{\partial\mathbf{W}(k,:)}$, and $\frac{\partial\mathcal{L}}{\partial\mathbf{V}(j,:)}$ from the loss function (Eq. (8)).

Before describing our update procedure, we show that rows of a factor matrix have different inverse terms depending on the sparsity patterns of a slice matrix when updating a factor matrix. We compare our update rule with a naive update rule based on the derivative from the loss function (Eq. (1)) as shown in Fig. 3(c). Assume that we update the factor matrix $\mathbf{U}_k$. The naive update rule updates each row of $\mathbf{U}_k$ using Eq. (2), and all the rows of $\mathbf{U}_k$ are updated using the same inverse term $\left(\mathbf{S}_k\mathbf{V}^T\mathbf{V}\mathbf{S}_k\right)^{-1}$. In contrast, each row of $\mathbf{U}_k$ is updated based on Eq. (4) when we use our update rule. Considering the sparsity pattern for each row, we compute the inverse term differently for each row. For

Algorithm 1: ATOM

```
Input: $\mathbf{X}_k \in \mathbb{R}^{I_k \times J}$ for $k = 1, ..., K$
Output: $\mathbf{U}_k \in \mathbb{R}^{I_k \times R}$, $\mathbf{S}_k \in \mathbb{R}^{R \times R}$ for $k = 1, ..., K$, and $\mathbf{V} \in \mathbb{R}^{J \times R}$.
Parameters: target rank $R$
 1: initialize matrices $\mathbf{U}_k$, $\mathbf{Q}_k$, $\mathbf{S}_k$ for $k = 1, ..., K$, and $\mathbf{H}$ and $\mathbf{V}$
 2: repeat
 3:     for $k = 1, ..., K$ do
 4:         update $\mathbf{U}_k$, $\mathbf{W}$, and $\mathbf{V}$ using Eq. (9), (10), and (11), respectively
 5:         update $\mathbf{Q}_k$ and $\mathbf{H}$ using Eq. (12) and (13), respectively
 6:     end for
 7: until the maximum iteration is reached, or the error ceases to decrease;
 8: for $k = 1, ..., K$ do
 9:     $\mathbf{S}_k \leftarrow diag(\mathbf{W}(k, :))$
10: end for
```

example, let $\mathbf{X}_k \in \mathbb{R}^{2\times 3}$ be a slice matrix where $\mathbf{X}_k(1, 2)$, $\mathbf{X}_k(2, 1)$, and $\mathbf{X}_k(2, 3)$ are missing. The naive update rule updates both $\mathbf{U}_k(1, :)$ and $\mathbf{U}_k(2, :)$ using $\left(\mathbf{S}_k\mathbf{V}^T\mathbf{V}\mathbf{S}_k\right)^{-1}$. Based on our update rule, we update $\mathbf{U}_k(1, :)$ using the inverse term $\left(\mathbf{S}_k \left(\mathbf{V}(1, :)^T\mathbf{V}(1, :) + \mathbf{V}(3, :)^T\mathbf{V}(3, :)\right) \mathbf{S}_k\right)^{-1}$ while updating $\mathbf{U}_k(2, :)$ using $\left(\mathbf{S}_k\mathbf{V}(2, :)^T\mathbf{V}(2, :)\mathbf{S}_k\right)^{-1}$.

We propose an alternating optimization based update procedure which independently updates rows of a factor matrix while fixing the other factor matrices; it computes $\frac{\partial \mathcal{L}}{\partial \mathbf{U}_k(i,:)} = 0$ while fixing $\mathbf{S}_k$ and $\mathbf{V}$, and updates $\mathbf{U}_k(i, ;)$ by arranging the equation of the gradient. Other factor matrices, $\mathbf{S}_k$ and $\mathbf{V}$, are also updated in the same manner. The proofs of Lemmas 1 to 5 are described in Appendix.

Updating $\mathbf{U}_k$. Based on the loss function (Eq. (8)), we update $\mathbf{U}_k(i, :)$ by setting $\frac{\partial \mathcal{L}}{\partial \mathbf{U}_k(i,:)} = 0$ and arranging it. We update $\mathbf{U}_k$ row by row with the following lemma:

Lemma 1. *When fixing all the factor matrices except for $\mathbf{U}_k$, the following update for the ith row of the factor matrix $\mathbf{U}_k$ minimizes the loss function (Eq. (8)):*

$$\mathbf{U}_k(i, :) \leftarrow \Bigg(\mathbf{X}_k(i, :)\mathbf{V}\mathbf{S}_k + \lambda_u\mathbf{Q}_k(i, :)\mathbf{H} + \lambda_s(\mathbf{U}_k(i - 1, :) + \mathbf{U}_k(i + 1, :))\Bigg) \times \Bigg((\lambda_u + 2\lambda_s)\mathbf{I} + \sum_{(i,j)\in\Omega_{k,i}} \mathbf{S}_k\mathbf{V}(j, :)^T\mathbf{V}(j, :)\mathbf{S}_k\Bigg)^{-1} \quad (9)$$

where $\Omega_{k,i}$ is the observable entries of the ith row of the kth slice matrix. Updating the first and the last row of $\mathbf{U}_k$ uses λ_s instead of $2\lambda_s$, and does not entail computing $\mathbf{U}_k(i - 1, :)$ and $\mathbf{U}_k(i + 1, :)$, respectively. □

Updating $\mathbf{S}_k$. To update $\mathbf{S}_k$ for $k = 1...K$, we first transform $\mathbf{S}_k$ for $k = 1...K$ into $\mathbf{W} \in \mathbb{R}^{K\times R}$ whose kth row contains the diagonal elements of $\mathbf{S}_k$ (i.e., $\mathbf{W}(k, r) = \mathbf{S}_k(r, r)$). Then, we update $\mathbf{W}(k, :)$ by setting $\frac{\partial \mathcal{L}}{\partial \mathbf{W}(k,:)} = 0$ and arranging it based on the loss function (Eq. (8)). We update $\mathbf{W}$ row by row with the following lemma:

Lemma 2. *When all the factor matrices except for $\mathbf{W}$ are fixed, the following update for the kth row of the factor matrix $\mathbf{W}$ which corresponds to the diagonal elements of $\mathbf{S}_k$ minimizes the loss function (Eq. (8)):*

$$\mathbf{W}(k, :) \leftarrow \Big(vec(\mathbf{X}_k)^T(\mathbf{V} \odot \mathbf{U}_k)\Big) \times \Bigg(\lambda_l\mathbf{I} + \sum_{(i,j)\in\Omega_k} \mathbf{V}(j, :)^T\mathbf{V}(j, :) * \mathbf{U}_k(i, :)^T\mathbf{U}_k(i, :)\Bigg)^{-1} \quad (10)$$

where $\odot$ is Khatri-Rao product, $*$ is element-wise matrix multiplication, and $vec(\cdot)$ is a vectorization operation which transforms a matrix into a vector. □

Note that only factor vectors corresponding to observable entries are used in the inverse term. This leads to excluding missing values and accurately updating the factor matrix, $\mathbf{S}_k$.

Updating $\mathbf{V}$. Then, we update $\mathbf{V}(j, :)$ by setting $\frac{\partial \mathcal{L}}{\partial \mathbf{V}(j,:)} = 0$ and arranging it based on the loss function (Eq. (8)). The following lemma describes how to update the factor matrix $\mathbf{V}$ row by row:

Lemma 3. *When fixing all the factor matrices except for $\mathbf{V}$, the following update for the jth row of the factor matrix $\mathbf{V}$ minimizes the loss function (Eq. (8)):*

$$\mathbf{V}(j, :) \leftarrow \left(\sum_{k=1}^{K}\Big(\mathbf{X}_k(:, j)^T\mathbf{U}_k\mathbf{S}_k\Big)\right) \times \left(\lambda_l\mathbf{I} + \sum_{k=1}^{K}\sum_{(i,j)\in\Omega_{k,j}} \mathbf{S}_k\mathbf{U}_k(i, :)^T\mathbf{U}_k(i, :)\mathbf{S}_k\right)^{-1} \quad (11)$$

where $\Omega_{k,j}$ consists of indices of the observed entries whose column index is equal to j in Ω_k. □

Updating $\mathbf{Q}_k$ and $\mathbf{H}$. We update $\mathbf{Q}_k$ and $\mathbf{H}$, which are factor matrices for the uniqueness regularization, with the following lemmas. Since they are not directly related to missing values, we update the entire elements of the matrices at once.

Lemma 4. *When fixing all the factor matrices except for $\mathbf{Q}_k$, we update the factor matrix $\mathbf{Q}_k$ by solving Orthogonal Procrustes Problem [28] due to column-orthogonality:*

$$\mathbf{Q}_k \leftarrow \mathbf{Z}_k\mathbf{P}_k^T \quad (12)$$

where $\mathbf{Q}_k$ is a column-orthogonal matrix (i.e., $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$), and $\mathbf{Z}_k$ and $\mathbf{P}_k^T$ are left and right singular vector matrices of $\mathbf{U}_k\mathbf{H}^T$, respectively. □

Lemma 5. *When fixing all the factor matrices except for $\mathbf{H}$, the following update for the factor matrix $\mathbf{H}$ minimizes the loss function (Eq. (8)):*

$$\mathbf{H} \leftarrow \frac{1}{K}\sum_{k=1}^{K}\mathbf{Q}_k^T\mathbf{U}_k \quad (13)$$

where K is the number of slice matrices in a given irregular tensor. □

The overall procedure of ATOM is shown in Algorithm 1. We first initialize all factor matrices (line 1 in Algorithm 1). Then, we alternatively update factor matrices with our update procedure in Lemmas 1 to 5 (lines 4 and 5 in Algorithm 1). We repeat the updates until the stop condition is met (lines 2 to 7 in Algorithm 1) and then obtain $\mathbf{S}_k$ with $\mathbf{W}$ (line 10 in Algorithm 1).

TABLE II
DESCRIPTION OF REAL-WORLD IRREGULAR TENSOR DATA. # OF NNZ, K, M, AND B INDICATE THE NUMBER OF NONZEROS, THOUSANDS, MILLIONS, AND BILLIONS, RESPECTIVELY.

Dataset	Max Dim. I_k	Dim. J	Dim. K	# of nnz	Summary
US Stoctk[1] [23]	$7,883$	88	$4,742$	1.3B	Stock
Korea Stock[2] [14]	$5,270$	88	$3,619$	859M	Stock
China Stock	$2,431$	88	219	46M	Stock
Japan Stock	$2,204$	88	215	41M	Stock
VicRoads[3] [29]	$1,084$	96	$2,033$	197M	Traffic
PEMS-SF[4]	440	144	963	60M	Traffic
ML-100k[5]	40	$1,682$	344	50K	Movie rating

D. Convergence Analysis

We theoretically prove the convergence of ATOM.

Theorem 1 (Convergence of ATOM)**.** *The update procedure of* ATOM *decreases the loss function (Eq.* (8)*) monotonically under the condition that the inverse terms in Eq.* (9), (10), *and* (11) *exist.*

Proof. As seen in Lemmas 1 to 5 and their proofs described in Appendix, every update in ATOM never increases a loss with respect to each factor matrix. Therefore, loss values never increase, and ATOM converges. □

ATOM converges to the local minimum by monotonically decreasing the loss function (Eq. (8)). We provide experimental results for the convergence in Section IV-E. Our proposed update procedure does not guarantee convergence to the global minimum but works well in practice by generating accurate factor matrices while guaranteeing the local minimum.

IV. EXPERIMENTS

We present experimental results to answer the following questions:

Q1 **Performance (Section IV-B).** How accurately does ATOM predict missing values for real-world irregular tensors?

Q2 **Ablation Study (Section IV-C).** Do the ideas of temporal smoothness and handling of missing values contribute to the performance of ATOM?

Q3 **Hyperparameter Sensitivity (Section IV-D).** How much do the regularization hyperparameters affect the performance of ATOM?

Q4 **Converge (Section IV-E).** Does ATOM converge without increasing loss values?

A. Experimental Settings

We present our experimental settings: machine, datasets, competitors, task, evaluation metric, and hyperparameters.

Machine. We use a workstation with 2 CPUs (Intel Xeon E5-2630 v4 @ 2.2GHz) where each CPU has 10 cores and 512GB memory.

Real-world Data. Table II describes the summary of datasets used in our experiment. We use 7 real-world temporal irregular tensors. The first four datasets consist of stocks in the United States, South Korea, China, and Japan, respectively. For stock datasets, each temporal irregular tensor is a collection of matrices each of which corresponds to a stock. The form of each slice matrix is $(date, feature)$. The time interval is by day and the feature dimension includes 1) 5 basic features consisting of opening price, closing price, highest price, lowest price, and trading volume, and 2) 83 technical indicators. The 83 technical indicators are extracted using the Technical Analysis library[6] based on the 5 basic features. The next two datasets VicRoads and PEMS-SF are related to traffic volume. They are represented as temporal irregular tensors of the form $(date, timeframe, location)$; location corresponds to a slice matrix whose form is $(date, timeframe)$ where each timeframe represents a given range of time in the day with fixed length: 15 minutes for VicRoads and 10 minutes for PEMS-SF. The last dataset, ML-100k, is a collection of ratings of items by users represented as a temporal irregular tensor of the form $(time, movie, user)$; a user corresponds to a slice matrix, and its form is $(time, movie)$. The time intervals between rows are not necessarily the same since users watch and rate movies whenever they want. Note that we do not use the slice matrices whose row size is equal to 1.

Normalization. We normalize real-world datasets based on their characteristics. For the four stock datasets, we use z-normalization for each jth column $\mathbf{X}(:, j)$ of a slice matrix $\mathbf{X}$. For VicRoads and PEMS-SF datasets, we use min-max normalization for each jth column $\mathbf{X}(:, j)$ of a slice matrix $\mathbf{X}$ since these datasets contain only non-negative values and maintain this characteristic. For ML-100k data, we divide all values by 5 to make their ratings 0 to 1 since their range is from 1 to 5.

Competitor. We compare ATOM with PARAFAC2 decomposition methods working on irregular tensors: PARAFAC2-ALS, RD-ALS [21], DPar2 [23], Spartan [16], COPA [17], and REPAIR [18]. We use the codes of DPar2[7] [23], Spartan[8] [16], COPA[9] [17], and REPAIR[10] [18] provided by the authors. We implement the code of RD-ALS [21] since there is no public code. Spartan [16], COPA [17], and REPAIR [18] have the sparsity and non-negativity regularizations that focus on improving their interpretability under such constraints. Especially, COPA [17] and REPAIR [18] utilize the non-negativity constraint to implement temporal smoothness regularization, which is applicable only to datasets with no negative values. Therefore, we remove these regularizations that degrade the accuracy of missing value prediction, and then evaluate the performance of these competitors for US Stock, Korea Stock, China Stock, and Japan Stock datasets which contain negative values.

Task. To evaluate the performance of ATOM, we perform a missing value prediction task. We randomly split observed

[1]https://datalab.snu.ac.kr/dpar2
[2]https://github.com/jungijang/KoreaStockData
[3]https://github.com/florinsch/BigTrafficData
[4]http://www.timeseriesclassification.com/
[5]https://grouplens.org/datasets/movielens/
[6]https://technical-analysis-library-in-python.readthedocs.io/en/latest/
[7]https://datalab.snu.ac.kr/dpar2/
[8]https://github.com/kperros/SPARTan
[9]https://github.com/aafshar/COPA
[10]https://github.com/Emory-AIMS/Repair

TABLE III

Performance for missing value prediction. Note that bold and underlined fonts indicate the lowest and the second-lowest errors, respectively. Atom outperforms the competitors on all the experimental settings: datasets and missing value ratios.

Missing Value Prediction - Test Ratio: 10%							
Model	US Stock	Korea Stock	China Stock	Japan Stock	VicRoads	PEMS-SF	ML-100k
PARAFAC2-ALS	0.4148 ± 0.0018	0.4554 ± 0.0002	0.2620 ± 0.0016	0.2695 ± 0.0001	0.0958 ± 0.0011	0.1269 ± 0.0003	0.6124 ± 0.0037
RD-ALS [21]	0.4154 ± 0.0002	0.4508 ± 0.0001	$\underline{0.2584 \pm 0.0005}$	0.2682 ± 0.0001	0.0666 ± 0.0004	0.1204 ± 0.0010	0.6028 ± 0.0010
DPar2 [23]	0.4139 ± 0.0002	$\underline{0.4487 \pm 0.0004}$	0.2588 ± 0.0006	$\underline{0.2654 \pm 0.0003}$	$\underline{0.0637 \pm 0.0004}$	$\underline{0.1174 \pm 0.0004}$	0.6042 ± 0.0016
SPartan [16]	$\underline{0.4117 \pm 0.0005}$	0.4500 ± 0.0010	0.2600 ± 0.0021	0.2670 ± 0.0000	0.1049 ± 0.0020	0.1307 ± 0.0014	0.6080 ± 0.0022
COPA [17]	0.4178 ± 0.0018	0.4583 ± 0.0000	0.2738 ± 0.0015	0.2713 ± 0.0000	0.1123 ± 0.0003	0.2192 ± 0.0008	$\underline{0.6023 \pm 0.0019}$
REPAIR [18]	0.4547 ± 0.0052	0.4866 ± 0.0000	0.3136 ± 0.0044	0.3039 ± 0.0000	0.1225 ± 0.0007	0.2393 ± 0.0013	0.7562 ± 0.0010
Atom	$\mathbf{0.3677 \pm 0.0013}$	$\mathbf{0.3994 \pm 0.0003}$	$\mathbf{0.2086 \pm 0.0023}$	$\mathbf{0.2196 \pm 0.0007}$	$\mathbf{0.0365 \pm 0.0001}$	$\mathbf{0.0854 \pm 0.0002}$	$\mathbf{0.0778 \pm 0.0014}$

Missing Value Prediction - Test Ratio: 20%							
Model	US Stock	Korea Stock	China Stock	Japan Stock	VicRoads	PEMS-SF	ML-100k
PARAFAC2-ALS	0.4618 ± 0.0006	0.4894 ± 0.0011	0.3072 ± 0.0027	0.3154 ± 0.0008	0.1578 ± 0.0013	0.1719 ± 0.0006	0.6426 ± 0.0032
RD-ALS [21]	0.4630 ± 0.0005	0.4855 ± 0.0002	0.3044 ± 0.0003	0.3153 ± 0.0002	0.1041 ± 0.0006	0.1575 ± 0.0012	0.6312 ± 0.0013
DPar2 [23]	0.4610 ± 0.0002	$\underline{0.4831 \pm 0.0002}$	$\underline{0.3027 \pm 0.0005}$	$\underline{0.3130 \pm 0.0004}$	$\underline{0.0999 \pm 0.0003}$	$\underline{0.1552 \pm 0.0001}$	0.6305 ± 0.0011
SPartan [16]	$\underline{0.4598 \pm 0.0007}$	0.4867 ± 0.0009	0.3035 ± 0.0005	0.3142 ± 0.0005	0.1662 ± 0.0024	0.1757 ± 0.0018	0.6428 ± 0.0065
COPA [17]	0.4664 ± 0.0027	0.4922 ± 0.0007	0.3169 ± 0.0021	0.3164 ± 0.0000	0.1410 ± 0.0007	0.2383 ± 0.0008	$\underline{0.6283 \pm 0.0040}$
REPAIR [18]	0.5109 ± 0.0037	0.5322 ± 0.0025	0.3659 ± 0.0053	0.3535 ± 0.0000	0.1569 ± 0.0010	0.2732 ± 0.0015	0.7764 ± 0.0028
Atom	$\mathbf{0.3942 \pm 0.0022}$	$\mathbf{0.4147 \pm 0.0028}$	$\mathbf{0.2200 \pm 0.0026}$	$\mathbf{0.2357 \pm 0.0014}$	$\mathbf{0.0374 \pm 0.0001}$	$\mathbf{0.0912 \pm 0.0001}$	$\mathbf{0.0798 \pm 0.0022}$

entries of a given data into training and test entries with the following ratios: (90%, 10%), (80%, 20%), and (70%, 30%). Note that ML-100k dataset has about 1% observed entries among all entries while almost all of the elements are observed in the other datasets. We learn factor matrices using training entries of irregular tensors, and predict values of test entries using the learned factor matrices.

Normalized Reconstruction Error. For each irregular tensor, we measure test errors by the normalized reconstruction error (NRE) defined as follows:

$$\text{NRE} = \left(\frac{\sum_{k=1}^{K} \sum_{(i,j)\in\Omega_{k,test}} \left(\mathbf{X}_k(i,j) - \hat{\mathbf{X}}_k(i,j) \right)^2}{\sum_{k=1}^{K} \sum_{(i,j)\in\Omega_{k,test}} \left(\mathbf{X}_k(i,j) \right)^2} \right)$$

where $\Omega_{k,test}$ includes test entries of the kth slice matrix, $\mathbf{X}_k$ is the k-th input slice, and $\hat{\mathbf{X}}_k$ is the k-th reconstructed slice of PARAFAC2 decomposition. Low NRE indicates that a method decomposes a tensor well.

Hyperparameters. We use several hyperparameters for Atom: target rank, and regularization strengths λ_s, λ_u, and λ_l. Except in Section IV-D, we set target rank R, λ_s, and λ_u to 10, 10, and 0.01, respectively. We set λ_l to 10 for ML-100k data while setting λ_l to 0 for the other datasets. The reason is to avoid computing the inverse of a singular matrix in ML-100k data where a few columns of slice matrices and a few slice matrices have a smaller number of training entries than R. The other datasets have a sufficient number of entries and thus they do not have the problem.

B. Performance

We compare Atom with competitors on the missing value prediction task while increasing the missing value test ratio from 10% to 30%. As shown in Fig. 2 and Table III, Atom gives the best performance for all datasets by up to 7.9× lower error rate than competitors. For ML-100k dataset, the performance gap is very large since this dataset has plenty of missing values consisting of unobserved entries and test entries, and existing methods fail to exclude missing values in updating factor matrices. In addition, the performance gap between Atom and the competitors grows as the test ratio increases from 10% to 30%. This is because handling missing values becomes more challenging for the competitors when we increase the test ratio.

C. Ablation Study

We provide an ablation study with respect to handling missing values and temporal smoothness. Atom-M and Atom-S correspond to Atom without the consideration of missing values and Atom without the temporal smoothness regularization, respectively. Atom-M-S is the same as PARAFAC2-ALS. Table IV shows that our update procedure with the loss function of the scalar form and temporal smoothness effectively reduces prediction errors. Atom-S, Atom without temporal smoothness, is not as good as Atom, but it has better performance than Atom-M-S and competitors (see Table III for comparisons). This result indicates that our update procedure finds accurate factor matrices in a temporal irregular tensor including missing values even without temporal smoothness.

D. Hyperparameter Sensitivity

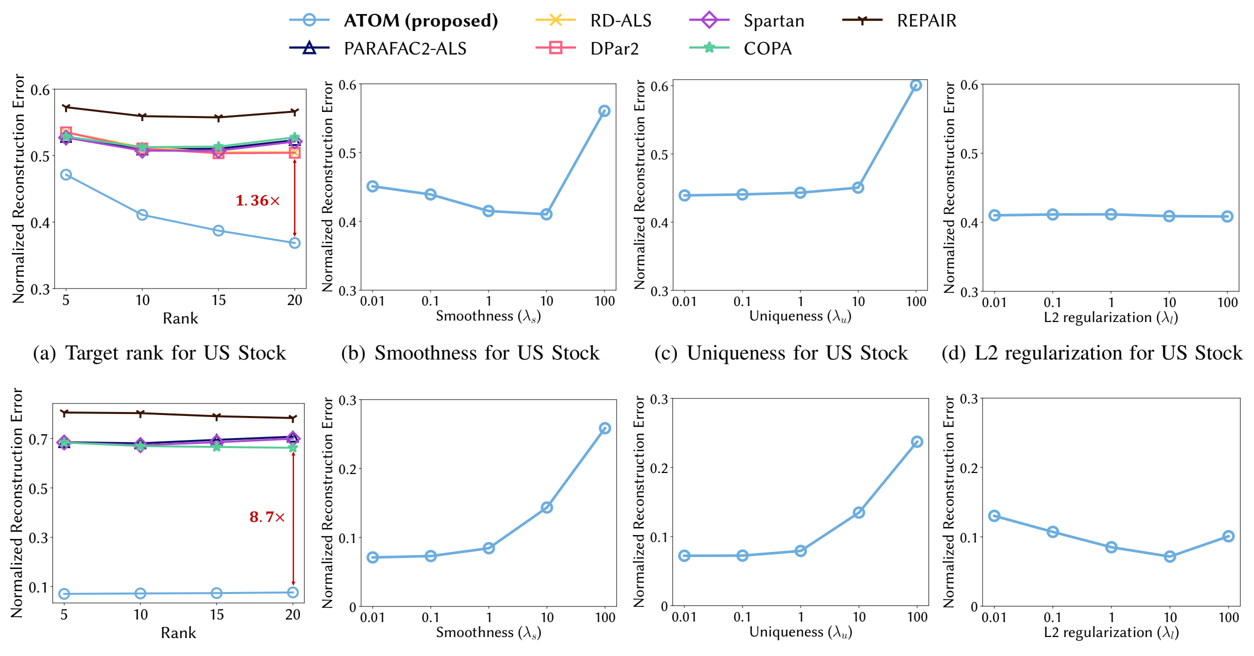
We evaluate the hyperparameter sensitivity of Atom by measuring prediction errors while varying hyperparameters: target rank, temporal smoothness, uniqueness, and L2 regularization. We use 4 target ranks $R \in [5, 10, 15, 20]$, 5 temporal smoothness hyperparameters $\lambda_s \in [0.01, 0.1, 1, 10, 100]$, 5 uniqueness hyperparameters $\lambda_u \in [0.01, 0.1, 1, 10, 100]$, and 5 L2 regularization hyperparameters $\lambda_l \in [0.01, 0.1, 1, 10, 100]$.

Rank. For the four ranks 5, 10, 15, and 20, we compare Atom with competitors on the two datasets, US Stock and ML-100k datasets. As shown in Fig. 5(a), Atom outperforms the competitors in US Stock while achieving up to 1.36× lower error rate than existing PARAFAC2 decomposition methods on a large target rank $R = 20$. In contrast to the competitors that fail to reduce the errors in the large target rank, Atom clearly decreases the errors as target rank increases. As shown in Fig. 5(e), Atom also outperforms the

TABLE IV

ABLATION STUDY. -S AND -M INDICATE THE ELIMINATION OF TEMPORAL SMOOTHNESS REGULARIZATION AND NO CONSIDERATION OF MISSING VALUES, RESPECTIVELY. BOLD AND UNDERLINED FONTS INDICATE THE LOWEST AND THE SECOND-LOWEST ERRORS, RESPECTIVELY. NOTE THAT BOTH THE TEMPORAL SMOOTHNESS AND CONSIDERATION OF MISSING VALUES CONTRIBUTE TO IMPROVING THE PREDICTION ACCURACY.

Missing Value Prediction - Missing Ratio: 30%							
Model	US Stock	Korea Stock	China Stock	Japan Stock	VicRoads	PEMS-SF	ML-100k
ATOM- M - S	0.5046 ± 0.0017	0.5341 ± 0.0012	0.5206 ± 0.0012	0.5141 ± 0.0006	0.2171 ± 0.0007	0.2228 ± 0.0006	0.6757 ± 0.0011
ATOM- M	0.4956 ± 0.0008	0.5252 ± 0.0014	0.5136 ± 0.0019	0.5066 ± 0.0009	0.2157 ± 0.0012	0.2213 ± 0.0005	0.6798 ± 0.0021
ATOM- S	0.4533 ± 0.0042	0.4822 ± 0.0040	0.4571 ± 0.0013	0.4510 ± 0.0010	0.0409 ± 0.0001	0.0986 ± 0.0001	0.1268 ± 0.0013
ATOM	$\mathbf{0.4094 \pm 0.0027}$	$\mathbf{0.4399 \pm 0.0017}$	$\mathbf{0.4269 \pm 0.0021}$	$\mathbf{0.4210 \pm 0.0008}$	$\mathbf{0.0391 \pm 0.0001}$	$\mathbf{0.0976 \pm 0.0001}$	$\mathbf{0.1005 \pm 0.0016}$



(a) Target rank for US Stock (b) Smoothness for US Stock (c) Uniqueness for US Stock (d) L2 regularization for US Stock

(e) Target rank for ML-100k (f) Smoothness for ML-100k (g) Uniqueness for ML-100k (h) L2 regularization for ML-100k

Fig. 5. Hyperparameter sensitivity in terms of normalized reconstruction error on US Stock and ML-100k datasets. See Section IV-D for details.

competitors in ML-100k while achieving up to 8.7× lower error rate than existing PARAFAC2 decomposition methods. Since the previous methods set missing values to zero, they fail to predict them accurately. ATOM has similar errors for all the ranks since the number of nonzero values in ML-100k is insufficient for learning large factor matrices (e.g., target ranks 15 and 20).

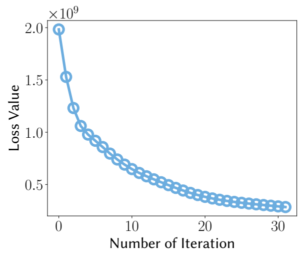
Smoothness. We evaluate the temporal smoothness regularization of ATOM. As shown in Fig. 5(b) and 5(f), the temporal smoothness is more effective on US Stock dataset than on ML-100k dataset; a large λ_s (e.g., 10) provides lower error than a small λ_s for US Stock dataset while small λ_s provides lower error than large λ_s for ML-100k dataset. This is because there is a clear temporal pattern where the value changes smoothly over time on the US stock dataset. A slice matrix in US Stock dataset has a constant time interval (e.g., day) while a slice matrix of a user in ML-100k dataset has irregular time intervals.

Uniqueness. We measure the prediction performance with respect to uniqueness regularization. As shown in Fig. 5(c) and 5(g), NRE of ATOM are proportional to the uniqueness hyperparameter λ_u. The uniqueness regularization degrades prediction performance although it helps improve the interpretability for factor matrices as used in previous methods [16]–[18].

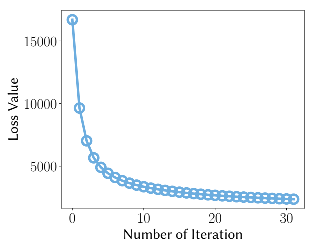
L2 regularization. We measure the prediction performance concerning L2 regularization. As shown in Fig. 5(d) and 5(h), there are different patterns between the two datasets. For US Stock dataset, the errors do not vary significantly with respect to the hyperparameter of L2 regularization. Since there are plenty of observed values in US Stock dataset for updating factor matrices, $\mathbf{W}$ and $\mathbf{V}$ are stably updated even without L2 regularization. For ML-100k dataset, L2 regularization is effective since the number of observed values is small. L2 regularization provides stable updates by avoiding computations of the inverse of a singular matrix and overfitting.

E. Convergence

For US Stock and ML-100k datasets, we experimentally evaluate the convergence of ATOM. We measure the loss value of (Eq. (8)) at each iteration. Fig. 6 shows that the loss values dramatically decrease in the first 5 iterations and locally converge in the subsequent iterations. This result is consistent with the theoretical analysis in Section III-D that ATOM converges to a minimum where loss values do not increase.



(a) US Stock



(b) ML-100k

Fig. 6. Convergence in updating factor matrices of Atom.

V. Related Works

We review previous works on PARAFAC2 decomposition for irregular tensors and their relation to ours.

PARAFAC2 decomposition for irregular tensors. Many works have proposed efficient PARAFAC2 methods to interpret irregular tensors, as PARAFAC2 is the only practical method for irregular tensor decomposition as of now. In contrast to the classical ALS algorithm for PARAFAC2 decomposition, RD-ALS [21] and DPar2 [23] involve a preprocessing stage of a given tensor before updating the factor matrices. SPADE [30] is an effective method designed to handle irregular tensors in online streaming, even for higher-order tensors. These methods are limited in handling real-world tensors with missing data since they treat missing values to zero. Perros et al. proposed SPARTan [16], a scalable PARAFAC2 decomposition method for irregular sparse tensors, and later works such as COPA [17] and REPAIR [18] have improved its performance and robustness with additional constraints. They provide phenotype discoveries by obtaining interpretable factor matrices, but they also fail to provide accurate factor matrices of PARAFAC2 decomposition due to setting the value of missing entries to zeros.

Applications of PARAFAC2 decomposition. Previous works [31], [32] suggest the application of PARAFAC2 for monitoring substances and multi-way analysis in chemometrics which extracts information from chemical systems. Many recent works have used PARAFAC2 decomposition for analyzing EHR data which can be represented as irregular sparse tensors with many missing entries. Perros et al. [33] applied PARAFAC2 decomposition on EHR data of medically complex children to discover phenotypes and temporal trends. TASTE [34] proposed a coupled irregular tensor decomposition for EHR data. LogPar [20] is a logistic PARAFAC2 decomposition method for binary irregular tensors representing temporal binary data including missing values. TedPar [35] considers the temporal dependency for better interpretation of EHR data. The above studies exploit the irregularity of one mode in multi-dimensional data, and maximize the effect of PARAFAC2 decomposition.

Atom is a novel PARAFAC2 decomposition method which accurately handles temporal irregular tensors with missing values.

VI. Conclusion

In this paper, we propose Atom, an accurate PARAFAC2 decomposition method for temporal irregular tensors with missing values. We observe that existing PARAFAC2 decomposition methods fail to fully exclude missing values in updating factor matrices. To address the issue, we reformulate the loss function for PARAFAC2 decomposition and propose a row by row update rule that considers the sparsity patterns of rows of a temporal irregular tensor. In addition, we further improve the accuracy by adding effective temporal smoothness regularization. Extensive experiments show that Atom outperforms existing PARAFAC2 decomposition methods providing up to 7.9× lower error rate. Future works include extending Atom to predicting future values and speeding up Atom while maintaining accuracy.

Acknowledgment

This work was supported by the National Research Foundation of Korea(NRF) funded by MSIT(2022R1A2C3007921). This work was also supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) [No.2022-0-00641, XVoice: Multi-Modal Voice Meta Learning], [No.2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)], and [NO.2021-0-02068, Artificial Intelligence Innovation Hub (Artificial Intelligence Institute, Seoul National University)]. The Institute of Engineering Research at Seoul National University provided research facilities for this work. The ICT at Seoul National University provides research facilities for this study. U Kang is the corresponding author.

References

[1] O. A. Malik and S. Becker, "Low-rank tucker decomposition of large tensors using tensorsketch," in *NeurIPS*, 2018.
[2] J. Jang and U. Kang, "D-tucker: Fast and memory-efficient tucker decomposition for dense tensors," in *ICDE*. IEEE, 2020, pp. 1850–1853.
[3] S. Oh, N. Park, J. Jang, L. Sael, and U. Kang, "High-performance tucker factorization on heterogeneous platforms," *TPDS*, vol. 30, no. 10, pp. 2237–2248, 2019.
[4] J.-G. Jang and U. Kang, "Static and streaming tucker decomposition for dense tensors," *TKDD*, 2022.
[5] D. Choi, J.-G. Jang, and U. Kang, "S3 cmtf: Fast, accurate, and scalable method for incomplete coupled matrix-tensor factorization," *PloS one*, vol. 14, no. 6, p. e0217316, 2019.
[6] K. Shin, L. Sael, and U. Kang, "Fully scalable methods for distributed tensor factorization," *TKDE*, vol. 29, no. 1, pp. 100–113, 2017.
[7] C. E. Tsourakakis, "MACH: fast randomized tensor decompositions," in *SDM*. SIAM, 2010, pp. 689–700.
[8] I. Jeon, E. E. Papalexakis, U. Kang, and C. Faloutsos, "Haten2: Billion-scale tensor decompositions," in *ICDE*. IEEE, 2015.
[9] S. Oh, N. Park, L. Sael, and U. Kang, "Scalable tucker factorization for sparse tensors - algorithms and discoveries," in *ICDE*. IEEE, 2018.
[10] M. R. de Araujo, P. M. P. Ribeiro, and C. Faloutsos, "Tensorcast: Forecasting with context using coupled tensors," in *ICDM*. IEEE, 2017.
[11] D. Lee and K. Shin, "Robust factorization of real-world tensor streams with patterns, missing values, and outliers," in *ICDE*. IEEE, 2021.
[12] X. Zhang, G. Wen, and W. Dai, "A tensor decomposition-based anomaly detection algorithm for hyperspectral image," *IEEE Trans. Geosci. Remote. Sens.*, vol. 54, no. 10, pp. 5801–5820, 2016.
[13] T. Kwon, I. Park, D. Lee, and K. Shin, "Slicenstitch: Continuous CP decomposition of sparse tensor streams," in *ICDE*. IEEE, 2021.
[14] J. Jang and U. Kang, "Fast and memory-efficient tucker decomposition for answering diverse time range queries," in *KDD*. ACM, 2021.

[15] T. Lacroix, G. Obozinski, and N. Usunier, "Tensor decompositions for temporal knowledge base completion," in *ICLR*. OpenReview.net, 2020.

[16] I. Perros, E. E. Papalexakis, F. Wang, R. W. Vuduc, E. Searles, M. Thompson, and J. Sun, "Spartan: Scalable PARAFAC2 for large & sparse data," in *SIGKDD*. ACM, 2017.

[17] A. Afshar, I. Perros, E. E. Papalexakis, E. Searles, J. C. Ho, and J. Sun, "COPA: constrained PARAFAC2 for sparse & large datasets," in *CIKM*. ACM, 2018.

[18] Y. Ren, J. Lou, L. Xiong, and J. C. Ho, "Robust irregular tensor factorization and completion for temporal health data analysis," in *CIKM*. ACM, 2020.

[19] N. E. Helwig, "Estimating latent trends in multivariate longitudinal data via parafac2 with functional and structural constraints," *Biometrical Journal*, vol. 59, no. 4, pp. 783–803, 2017.

[20] K. Yin, A. Afshar, J. C. Ho, W. K. Cheung, C. Zhang, and J. Sun, "Logpar: Logistic parafac2 factorization for temporal binary data with missing values," in *SIGKDD*, 2020.

[21] Y. Cheng and M. Haardt, "Efficient computation of the PARAFAC2 decomposition," in *ACSCC*. IEEE, 2019.

[22] R. A. Harshman, "Parafac2: Mathematical and technical notes," *UCLA working papers in phonetics*, vol. 22, no. 3044, 1972.

[23] J. Jang and U. Kang, "Dpar2: Fast and scalable parafac2 decomposition for irregular dense tensors," in *ICDE*. IEEE, 2022.

[24] D. Ahn, J. Jang, and U. Kang, "Time-aware tensor decomposition for sparse tensors," *Mach. Learn.*, vol. 111, no. 4, pp. 1409–1430, 2022.

[25] J. M. ten Berge and H. A. Kiers, "Some uniqueness results for parafac2," *Psychometrika*, vol. 61, no. 1, pp. 123–132, 1996.

[26] H. A. Kiers, J. M. Ten Berge, and R. Bro, "Parafac2—part i. a direct fitting algorithm for the parafac2 model," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 13, no. 3-4, 1999.

[27] R. A. Harshman and M. E. Lundy, "Uniqueness proof for a family of models sharing features of tucker's three-mode factor analysis and parafac/candecomp," *Psychometrika*, vol. 61, no. 1, pp. 133–154, 1996.

[28] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013, vol. 3.

[29] F. Schimbinschi, X. V. Nguyen, J. Bailey, C. Leckie, H. Vu, and R. Kotagiri, "Traffic forecasting in complex urban networks: Leveraging big data and machine learning," in *Big Data*. IEEE, 2015.

[30] E. Gujral, G. Theocharous, and E. E. Papalexakis, "Spade: S treaming pa rafac2 de composition for large datasets," in *SDM*. SIAM, 2020.

[31] S. Matero, S. Poutiainen, J. Leskinen, S.-P. Reinikainen, J. Ketolainen, K. Järvinen, and A. Poso, "Monitoring the wetting phase of fluidized bed granulation process using multi-way methods: The separation of successful from unsuccessful batches," *Chemometrics and Intelligent Laboratory Systems*, vol. 96, no. 1, pp. 88–93, 2009.

[32] N. Robeyst, C. U. Grosse, and N. De Belie, "Monitoring fresh concrete by ultrasonic transmission measurements: exploratory multi-way analysis of the spectral information," *Chemometrics and Intelligent Laboratory Systems*, vol. 95, no. 1, pp. 64–73, 2009.

[33] I. Perros, E. E. Papalexakis, R. Vuduc, E. Searles, and J. Sun, "Temporal phenotyping of medically complex children via parafac2 tensor factorization," *Journal of Biomedical Informatics*, vol. 93, 2019.

[34] A. Afshar, I. Perros, H. Park, C. Defilippi, X. Yan, W. Stewart, J. Ho, and J. Sun, "Taste: Temporal and static tensor factorization for phenotyping electronic health records," in *CHIL*, 2020.

[35] K. Yin, W. K. Cheung, B. C. Fung, and J. Poon, "Tedpar: Temporally dependent parafac2 factorization for phenotype-based disease progression modeling," in *SDM*. SIAM, 2021.

APPENDIX

We provide proofs for Lemmas 1 to 5 described in Section III-C.

A. Proof of Lemma 1

Proof. We first compute $\frac{\partial \mathcal{L}}{\partial \mathbf{U}_k(i,:)}$ as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}_k(i,:)} = -2 \sum_{(i,j)\in\Omega_{k,i}} \left(\left(\mathbf{X}_k(i,j) - \mathbf{U}_k(i,:)\mathbf{S}_k\mathbf{V}(j,:)^T \right) \mathbf{V}(j,:)\mathbf{S}_k \right)$$
$$- 2\lambda_s \left(\mathbf{U}_k(i-1,:) - \mathbf{U}_k(i,:) + \mathbf{U}_k(i+1,:) - \mathbf{U}_k(i,:) \right)$$
$$- 2\lambda_u \left(\mathbf{Q}_k(i,:)\mathbf{H} - \mathbf{U}_k(i,:) \right) \tag{14}$$

We set $\frac{\partial \mathcal{L}}{\partial \mathbf{U}_k(i,:)}$ to zero, arrange the equation for $\mathbf{U}_k(i,:)$, and obtain Eq. (9). Note that $\sum_{(i,j)\in\Omega_{k,i}} (\mathbf{X}_k(i,j))\,\mathbf{V}(j,:)\mathbf{S}_k$ is equal to $\mathbf{X}_k(i,:)\mathbf{V}\mathbf{S}_k$ when missing values in $\mathbf{X}_k(i,:)$ are treated as zeros. □

B. Proof of Lemma 2

Proof. We first compute $\frac{\partial \mathcal{L}}{\partial \mathbf{W}(k,:)}$ as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}(k,:)} = -2 \sum_{(i,j)\in\Omega_k} \Bigg(\left(\mathbf{X}_k(i,j) - \mathbf{W}(k,:)\left(\mathbf{V}(j,:)\odot\mathbf{U}_k(i,:)\right)^T \right)$$
$$\times \left(\mathbf{V}(j,:)\odot\mathbf{U}_k(i,:)\right) \Bigg) + 2\lambda_l \mathbf{W}(k,:) \tag{15}$$

We set $\frac{\partial \mathcal{L}}{\partial \mathbf{W}(k,:)}$ to zero, arrange the equation for $\mathbf{W}(k,:)$, and obtain Eq. (10). Note that $\sum_{(i,j)\in\Omega_k} (\mathbf{X}_k(i,j))\left(\mathbf{V}(j,:)\odot\mathbf{U}_k(i,:)\right)$ is equal to $\left(vec(\mathbf{X}_k)^T(\mathbf{V}\odot\mathbf{U}_k)\right)$ when missing values in $\mathbf{X}_k$ are treated as zeros. □

C. Proof of Lemma 3

Proof. We first compute $\frac{\partial \mathcal{L}}{\partial \mathbf{V}(j,:)}$ as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{V}(j,:)} = 2\lambda_l \mathbf{V}(j,:)$$
$$- 2\sum_{k=1}^{K} \sum_{(i,j)\in\Omega_{k,j}} \left(\left(\mathbf{X}_k(i,j) - \mathbf{V}(j,:)\mathbf{S}_k\mathbf{U}_k(i,:)^T \right) \mathbf{U}_k(i,:)\mathbf{S}_k \right) \tag{16}$$

We set $\frac{\partial \mathcal{L}}{\partial \mathbf{V}(j,:)}$ to zero, arrange the equation for $\mathbf{V}(j,:)$, and obtain Eq. (11). Note that $\sum_{i\in} (\mathbf{X}_k(i,j)\mathbf{U}_k(i,:)\mathbf{S}_k)$ is equal to $\mathbf{X}_k(:,j)^T\mathbf{U}_k\mathbf{S}_k$ when missing values in $\mathbf{X}_k$ are treated as zeros. □

D. Proof of Lemma 4

Proof. Minimizing $\|\mathbf{U}_k - \mathbf{Q}_k\mathbf{H}\|_F^2$ with respect to $\mathbf{Q}_k$ is equal to maximizing the following term:

$$tr(\mathbf{Q}_k^T\mathbf{U}_k\mathbf{H}^T) = < \mathbf{Z}_k^T\mathbf{Q}_k\mathbf{P}_k, \mathbf{\Sigma}_k >_F \tag{17}$$

where $\mathbf{Z}_k\mathbf{\Sigma}_k\mathbf{P}^T$ is the SVD result of $\mathbf{U}_k\mathbf{H}^T$, and tr and $< \cdot >_F$ indicate the trace and Frobenius inner product, respectively. Since $\mathbf{Z}_k$ and $\mathbf{Q}_k\mathbf{P}_k$ are column orthogonal matrices, $< \mathbf{Z}_k^T\mathbf{Q}_k\mathbf{P}_k, \mathbf{\Sigma}_k >_F$ is maximized when $\mathbf{Q}_k\mathbf{P}_k = \mathbf{Z}_k$ (i.e., $\mathbf{Z}_k^T\mathbf{Q}_k\mathbf{P}_k = \mathbf{I}$). Therefore, $\mathbf{Q}_k$ is updated using $\mathbf{Z}_k\mathbf{P}_k^T$ as in Eq. (12). □

E. Proof of Lemma 5

Proof. We compute $\frac{\partial \mathcal{L}}{\partial \mathbf{H}}$ as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{H}} = \sum_{k=1}^{K} \left(\mathbf{Q}_k^T(\mathbf{U}_k - \mathbf{Q}_k\mathbf{H}) \right) \tag{18}$$

We set $\frac{\partial \mathcal{L}}{\partial \mathbf{H}}$ to zero, arrange the equation for $\mathbf{H}$, and obtain Eq. (13). Note that $\mathbf{Q}_k^T\mathbf{Q}_k$ is equal to $\mathbf{I}$ since $\mathbf{Q}_k$ is a column-orthogonal matrix. □