Computational and Applied Mathematics (2023) 42:234
https://doi.org/10.1007/s40314-023-02368-w

TR-STF: a fast and accurate tensor ring decomposition algorithm via defined scaled tri-factorization

Ting Xu[1] · Ting-Zhu Huang[1] · Liang-Jian Deng[1] · Hong-Xia Dou[2] · Naoto Yokoya[3,4]

Received: 2 March 2023 / Revised: 29 May 2023 / Accepted: 9 June 2023 /
Published online: 27 June 2023

Abstract
This paper proposes an algorithm based on defined scaled tri-factorization (STF) for fast and accurate tensor ring (TR) decomposition. First, based on the fast tri-factorization approach, we define STF and design a corresponding algorithm that can more accurately represent various matrices while maintaining a similar level of computational time. Second, we apply sequential STFs to TR decomposition with theoretical proof and propose a stable (i.e., non-iterative) algorithm named TR-STF. It is a computationally more efficient algorithm than existing TR decomposition algorithms, which is beneficial when dealing with big data. Experiments on multiple randomly simulated data, highly oscillatory functions, and real-world data sets verify the effectiveness and high efficiency of the proposed TR-STF. For example, on the Pavia University data set, TR-STF is nearly 9240 and 39 times faster, respectively, and more accurate than algorithms based on alternating least squares and singular value decomposition. As an extension, we apply sequential STFs to tensor train (TT) decomposition and propose

✉ Ting-Zhu Huang
tingzhuhuang@126.com

✉ Liang-Jian Deng
liangjian.deng@uestc.edu.cn

Ting Xu
17742879536@163.com

Hong-Xia Dou
hongxiadou1991@126.com

Naoto Yokoya
yokoya@k.u-tokyo.ac.jp

1 School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu 611731, China

2 School of Science, XIHUA University, Chengdu 610039, China

3 The Department of Complexity Science and Engineering, Graduate School of Frontier Sciences, The University of Tokyo, Chiba 277-8561, Japan

4 Geoinformatics Unit, The RIKEN Center for Advanced Intelligence Project (AIP), Tokyo 103-0027, Japan

a non-iterative algorithm named TT-STF. Experimental results demonstrate the superiority of the proposed TT-STF compared with the state-of-the-art TT decomposition algorithm.

Keywords Tensor ring decomposition · Tensor train decomposition · Scaled tri-factorization · Fast algorithm

Mathematics Subject Classification 68W99

1 Introduction

For an dth-order tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, where $n_1 = n_2 = \cdots = n_d = n$, the storage cost (as well as the free parameters) of $\mathcal{X}$ grows exponentially as d increases. This exponential growth is commonly referred to as the *curse of dimensionality*, which makes it impractical to explicitly store all the entries of the tensor except for the minimal value of d. Even for $n = 4$, storing a tensor of order $d = 25$ would require 9 petabytes. Therefore, it is crucial to approximate higher-order tensors with compression schemes that do not destroy the inherent characteristics of tensors as much as possible, for example, low-rank tensor decomposition (Sultonov et al. 2023; Wang and Yang 2022; Cichocki et al. 2016; Xue et al. 2021b; Wang et al. 2022). Tensor decomposition, which represents a high-order tensor (i.e., $\geq$ 3rd-order tensor), as factor tensors and matrices, has been developed and applied in many fields, such as computer vision (Fu et al. 2019; Deng et al. 2019; Xiao et al. 2022; Tai et al. 2021; Xue et al. 2021a, 2022; Wang et al. 2023; Xu et al. 2022; Sun et al. 2023) and machine learning (Deng et al. 2021, 2022, 2023; Cao et al. 2020; Ran et al. 2023). The most well-known tensor decompositions include Canonical Polyadic (CP) decomposition (Bro 1997; De Lathauwer 2006; Jiang et al. 2022; Bozorgmanesh and Hajarian 2022; Xue et al. 2019) and Tucker decomposition (Tucker 1996; De Lathauwer et al. 2000; Luan et al. 2023; Che et al. 2021; Che and Wei 2020; Xue et al. 2020). Specifically, CP decomposition represents a tensor with a set of rank-1 tensors, where the smallest number of the rank-1 tensors denotes CP rank. Consequently, the storage cost of CP decomposition grows linearly with the tensor order. Assuming that the CP rank of the dth-order tensor with each dimension of n is m, then the storage cost of CP decomposition is nmd. However, there are no additional structural assumptions (De et al. 2016; Xu 2016; Qi et al. 2015; Brachat et al. 2010), (1) the estimation of the CP rank is an NP-hard problem (Hillar and Lim 2013), (2) the estimation of the rank-1 tensors is an ill-posed problem (DE Silva and Lim 2008), and (3) it is difficult to characterize different correlations among different modes flexibly (Zhao et al. 2016). These lead to (1) there is no guarantee of the existence of a low-rank approximation of a tensor through its rank-m CP decomposition, (2) a stable and efficient algorithm to estimate the rank-1 tensors may not exist, or it has poor performance when dealing with big data tensors, and (3) it performs poorly in the structure preservation of tensors. Tucker decomposition expresses a tensor by one core tensor and the factor matrices in all modes; Tucker rank is based on the Tucker decomposition, where the tensor is unfolded along each mode, which inevitably destroys the inherent structure of the tensor. In addition, the need for storing the $m_1 \times m_2 \cdots \times m_d$ core tensor renders the Tucker decomposition increasingly unappealing as the tensor order d gets large. In particular, the storage cost of Tucker decomposition grows exponentially with its dimensions. Assume that the Tucker rank of $\mathcal{X}$ is $[m, m, \ldots, m]^T$, then the storage cost of Tucker decomposition is $dnm + m^d$.

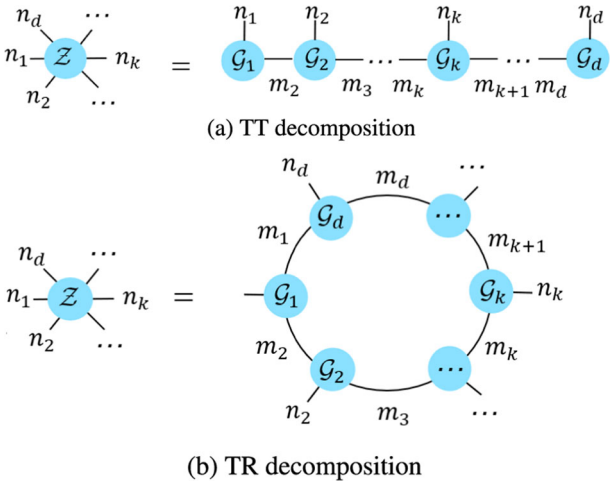

Fig. 1 Decomposition diagram of tensor $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \cdots \times n_d}$

Recently, tensor network, as an extension of tensor decomposition, has emerged as a means of decomposing a high-order tensor into sparsely interconnected small-scale cores and exhibited superior properties in compressing (Orus 2014; Huckle et al. 2013; Hashemizadeh et al. 2020; Ding et al. 2019; Zniyed et al. 2020). Particularly, some TN models are introduced to practical applications due to their excellent high-order data representation capabilities. Tensor train (TT) decomposition (Oseledets 2011; Holtz et al. 2012; Oseledets and Tyrtyshnikov 2010; Dektor et al. 2021) and tensor ring (TR) decomposition (Zhao et al. 2016) are two such networks. Specifically, as shown in Fig. 1a, TT decomposition tries to represent an dth-order tensor $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \cdots \times n_d}$ by a sequence of cores $\mathcal{G}_i \in \mathbb{R}^{m_i \times n_i \times m_{i+1}}$, $i = 1, 2, \ldots, d$, where $\mathbf{G}_1 \in \mathbb{R}^{n_1 \times m_2}$ and $\mathbf{G}_d \in \mathbb{R}^{m_d \times n_d}$ are matrices (i.e., $m_1 = m_{d+1} = 1$) and the remaining cores are 3rd-order tensors. Vector $[m_2, m_3, \ldots, m_d]^T$ is called the TT rank. TT decomposition plays an important role in complex tensor networks and has been successfully applied in many applications, e.g., hyperspectral image super-resolution and tensor completion (Dian et al. 2019; Ding et al. 2021). However, TT decomposition has some inevitable limitations. For example, the boundary condition, i.e., $m_1 = m_{d+1} = 1$, will limit its representation capacity and flexibility. To overcome these drawbacks, Zhao et al. (2016) propose TR decomposition, which represents $\mathcal{Z}$ by a sequence of cores $\mathcal{G}_k \in \mathbb{R}^{m_k \times n_k \times m_{k+1}}$, $k = 1, 2, \ldots, d$, where the size of the last and first cores satisfies $m_1 = m_{d+1}$, and vector $\mathbf{m} = [m_1, m_2, \ldots, m_d]^T$ is denoted as TR rank (see Fig. 1b). Compared with TT decomposition, TR decomposition keeps the invariance when the tensor dimension makes a circular shift. Thus, TR decomposition can better represent tensors flexibly with the same parameters as in TT decomposition. Given an dth-order tensor $\mathcal{Z} \in \mathbb{R}^{n \times n \times \cdots \times n}$, the TR rank and TT rank are $[m_1, m_2 \ldots, m_d]^T$ and $[m_2, m_3 \ldots, m_d]^T$, respectively, where $m_1 = m_2 = \cdots = m_d = m$. Then, the storage cost of TR decomposition and TT decomposition is dnm^2, which means that the storage cost of TR decomposition grows linearly with the tensor order d. Therefore, the TR decomposition helps to break the curse of dimensionality (Oseledets and Tyrtyshnikov 2009). In fact, TR decomposition degrades to TT decomposition when $m_1 = m_{d+1} = 1$. Based on its strong representation capacity, TR decomposition has been widely used in many applications and has produced competitive outputs, e.g., hyperspectral image denoising, hyperspectral image

compressive sensing, and remote sensing image reconstruction (Chen et al. 2020; He et al. 2019, 2022).

Given the size of input data, a stable and efficient algorithm for TR decomposition is critical in this era of big data. The current approaches mainly employ singular value decomposition (SVD) and alternating least squares (ALS) algorithms. Specifically, Zhao et al. (2016) utilize sequential SVDs to compute the cores (named TR-SVD), in which the TR rank can be estimated with a prescribed relative error (RE).[1] Further, Zhao et al. propose three ALS-based iterative algorithms, where each of the cores is updated by solving a least-squares problem (Xiao et al. 2021; Gnanasekaran 2022). Compared to ALS-based algorithms, TR-SVD is a non-iterative algorithm, resulting in higher computational speed. This advantage arises from the non-iterative nature of TR-SVD, which eliminates the need for repeated iterations that are inherent in ALS-based algorithms. However, for high-order data, the efficiency of TR-SVD is not satisfactory because it involves expensive SVD computation of large matrices. In this paper, we propose a non-iterative algorithm called TR-STF, based on defined scaled tri-factorization (STF), for fast and accurate TR decomposition. We are inspired by the fact that the existing fast tri-factorization (FTF) is often unfeasible when presented with some matrices without low rank due to its strict condition. We relax the condition to make it widely used, define STF, and design a corresponding algorithm to represent data more accurately in an acceptable time growth range. We then apply sequential STFs to TR decomposition with theoretical proof and propose TR-STF. Extensive experiments show that compared with the current state-of-the-art algorithms, the proposed TR-STF has advantages in accuracy and efficiency, especially the latter. As an extension, we apply the sequential STFs to TT decomposition with theoretical proof and propose the TT-STF, a non-iterative algorithm. The contributions of this paper are as follows:

- We define STF and design a corresponding algorithm that accurately represents various matrices.
- We propose a non-iterative TR-STF algorithm for fast and accurate TR decomposition using sequential STFs with theoretical proof.
- Through experiments on multiple data and tasks, we demonstrate that the proposed TR-STF outperforms state-of-the-art TR decomposition algorithms in efficiency and accuracy, especially the former. Moreover, we propose TT-STF based on sequential STFs for fast and accurate TT decomposition.

We arrange the remaining part as follows: The related work is introduced in Sect. 2. We present the proposed STF and TR-STF algorithms in Sect. 3. Section 4 gives the experimental results. The application of STF in TT decomposition is shown in Sect. 5. Finally, we conclude this paper in Sect. 6.

2 Related work

In this section, we will introduce works related to this paper.

2.1 Work-related notations

We use z, $\mathbf{z}$, $\mathbf{Z}$, and $\mathcal{Z}$ to represent scalars, vectors, matrices, and tensors, respectively. The field of real numbers is denoted as $\mathbb{R}$. Given an dth-order tensor $\mathcal{Z} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, its

[1] This denotes the relative error between the given tensor and the approximated tensor.

Fig. 2 Schematic diagram of FTF of matrix $\mathbf{Z} \in \mathbb{R}^{n_1 \times n_2}$, where $m \ll \min(n_1, n_2)$

$(j_1, j_2, \ldots, j_d)$th element and $(j_1, \ldots, j_{i-1}, j_{i+1}, \ldots, j_d)$th mode-$i$ fibe are denoted by $\mathcal{Z}(j_1, j_2, \ldots, j_d)$, $\mathbf{z}(j_1, \ldots, j_{i-1}, j_{i+1}, \ldots, j_d)$, respectively, Frobenius norm is denoted by $\|\mathcal{Z}\|_F$. When the order of $\mathcal{Z}$ is 3, i.e., d=3, we denote $\mathbf{Z}(j, :, :)$, $\mathbf{Z}(:, j, :)$, and $\mathbf{Z}(:, :, j)$ as the jth horizontal, lateral, and frontal slice, respectively, for brevity, $\mathbf{Z}(:, j, :)$ is written as $\mathbf{Z}(j)$. The inner product of two nth-order vectors $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^n$ is defined as $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x(i)y(i)$; the outer product of two vectors $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^n$ is defined a real matrix of order $m \times n$ obtained by multiplying each element in $\mathbf{x}$ and each element in $\mathbf{y}$, denoted as $\mathbf{x} \circ \mathbf{y}$. Tr$(\cdot)$, vec$(\cdot)$, Rank $(\cdot)$, and $(\cdot)^T$ represent the trace, vectorization, rank, and transpose of the matrix, respectively. $\mathbf{Q} = \text{qr}(\mathbf{A})$ means QR decomposition of $\mathbf{A}$ of size $m \times n$, where $\mathbf{Q}$ is the orthogonal matrix of size $m \times m$. We denote eye$(m, n) \in \mathbb{R}^{m \times n}$ as a matrix whose elements on the main diagonal are 1 and the remaining elements are 0. We denote $\mathbf{I}$ as an identity matrix. The permuted tensors of $\mathcal{Z}$ are denoted as "permute$(\mathcal{Z}, [i_1, i_2, \ldots, i_d])$", therein $[i_1, i_2, \ldots, i_d]$ is a random permutations of $[1, 2, \ldots, d]$.

2.2 Fast tri-factorization (FTF)

In recent work, Liu et al. (2013) propose FTF to avoid the high computational complexity of SVD. Specifically, the authors decompose matrix $\mathbf{Z} \in \mathbb{R}^{n_1 \times n_2}$ into three factor matrices, whose dimensions are all much smaller than $\mathbf{Z}$, as shown in Definition 1 and Fig. 2.

Definition 1 [*FTF* (Liu et al. 2013)] For a given matrix $\mathbf{Z} \in \mathbb{R}^{n_1 \times n_2}$, FTF of $\mathbf{Z}$ is

$$\mathbf{Z} = \mathbf{ABC}, \tag{1}$$

where $\mathbf{B} \in \mathbb{R}^{m \times m}$; $\mathbf{A} \in \mathbb{R}^{n_1 \times m}$ is column orthogonal, i.e., $\mathbf{A}^T\mathbf{A} = \mathbf{I}$, $\mathbf{C} \in \mathbb{R}^{m \times n_2}$ is row orthogonal, i.e., $\mathbf{CC}^T = \mathbf{I}$; and m is an upper bound on the rank of $\mathbf{Z}$ and satisfies $m \ll \min(n_1, n_2)$.

The condition, $m \ll \min(n_1, n_2)$, means that FTF is only applicable to low-rank matrices. However, most data in the real world have complex structures and may not be of low rank. In that case, this condition is no longer suitable. For example, we select the first band of one image (*x0043*) with the size of 321 × 481 × 3 from the CBSD68 data set[2] to test FTF with different values of m, as shown in Fig. 3. The information of the recovered image is seriously lost when the value of m is minimal, such as 100. Less information in the recovered image is lost with the increase in the value of m, such as 200 and 250. Therefore, it is necessary to increase the value range of m. Especially, when $m = 321$, the recovered image is perfect. In addition, with the increase of m, the running time does not increase much. When m is 321, the time-consuming is only 0.004 s longer than when m is 100, but the change of RelCha[3]

[2] The details of this data set can be found at https://www.github.com/clausmichele/CBSD68-dataset.

[3] RelCha is defined by

$$\text{RelCha} = \frac{\|\mathcal{Z} - \mathcal{R}(\mathcal{Z})\|_F}{\|\mathcal{Z}\|_F}, \tag{2}$$

where $\mathcal{Z}$ denotes the original image and $\mathcal{R}(\mathcal{Z})$ is the reconstructed image. The smaller the RelCha, the better the result.

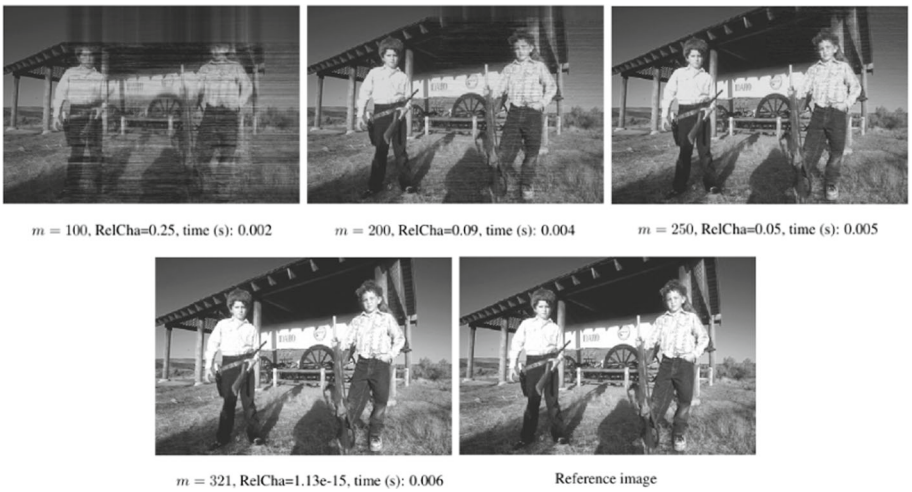

Fig. 3 Comparison of reconstructed image results of FTF under different m, where RelCha denotes the RE between the reference image and the reconstructed image

is obvious. Based on this observation, we define STF and design its algorithm as shown in detail in Sect. 3.1.

2.3 Tensor ring (TR) decomposition

This section introduces three forms of TR decomposition and gives several work-related definitions.

For an dth-order tensor $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \cdots \times n_d}$, the TR decomposition in the element-wise form is represented by

$$\begin{aligned} Z(j_1, j_2, \ldots, j_d) &= \mathrm{Tr}\{\mathbf{G}_1(j_1)\mathbf{G}_2(j_2)\ldots\mathbf{G}_d(j_d)\} \\ &= \mathrm{Tr}\left\{\prod_{i=1}^{d}\mathbf{G}_i(j_i)\right\}, \end{aligned} \tag{3}$$

where $\mathbf{G}_i(j_i) \in \mathbb{R}^{m_i \times m_{i+1}}$ is the j_ith lateral slice of the ith core $\mathcal{G}_i \in \mathbb{R}^{m_i \times n_i \times m_{i+1}}$. We rewrite (3) in the index form, i.e.,

$$Z(j_1, j_2, \ldots, j_d) = \sum_{\beta_1,\beta_2,\ldots,\beta_d=1}^{m_1,m_2,\ldots,m_d} \prod_{i=1}^{d} G_i(\beta_i, j_i, \beta_{i+1}), \tag{4}$$

where $\beta_{d+1} = \beta_1$ due to the trace operation. $\forall i \in \{1, \ldots, d\}$, $1 \le \beta_i \le m_i$, $1 \le j_i \le n_i$, where i is the index of tensor dimensions; β_k is the index of latent dimensions; and j_k is the index of data dimensions. Based on Eq. (4), we express TR decomposition in the tensor form, i.e.,

$$\mathcal{Z} = \sum_{\beta_1,\ldots,\beta_d=1}^{m_1,\ldots,m_d} \mathbf{g}_1(\beta_1, \beta_2) \circ \mathbf{g}_2(\beta_2, \beta_3) \circ \cdots \circ \mathbf{g}_d(\beta_d, \beta_1), \tag{5}$$

where the symbol "$\circ$" denotes the outer product of vectors and $\mathbf{g}_i(\beta_i, \beta_{i+1}) \in \mathbb{R}^{n_i}$ is the (β_i, β_{i+1})th mode-2 fiber of $\mathcal{G}_i$.

Definition 2 [*k-Unfolding Matrix* (Zhao et al. 2016)] Let $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \cdots \times n_d}$ be an dth-order tensor. The k-unfolding of $\mathcal{Z}$ is a matrix, denoted by $\mathbf{Z}_{\langle k \rangle}$ of size $\prod_{i=1}^{k} n_i \times \prod_{j=k+1}^{d} n_j$, whose elements obey

$$Z_{\langle k \rangle}(\overline{j_1 \dots j_k}, \overline{j_{k+1} \dots j_d}) = Z(j_1, j_2, \dots, j_d), \tag{6}$$

where the first k indices enumerate the rows of $\mathbf{Z}_{\langle k \rangle}$, and the last $d-k$ indices for its columns, the multi-indices $\overline{j_1 \dots j_k}$ and $\overline{j_{k+1} \dots j_d}$ are defined by $1 + \sum_{i=1}^{k} (j_i - 1) \prod_{l=1}^{i-1} n_l$ and $1 + \sum_{i=k+1}^{d} (j_i - 1) \prod_{l=k+1}^{i-1} n_l$, respectively.

Definition 3 [*Mode-k Unfolding Matrix* (Zhao et al. 2016)] Let $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \cdots \times n_d}$ be an dth-order tensor. The mode-k unfolding matrix of $\mathcal{Z}$ is denoted by $\mathbf{Z}_{[k]}$ of size $n_k \times \prod_{j \neq k}^{d} n_j$ with its elements defined by

$$Z_{[k]}(j_k, \overline{j_{k+1} \dots j_d j_1 \dots j_{k-1}}) = Z(j_1, j_2, \dots, j_d), \tag{7}$$

where kth indices enumerate the rows of $\mathbf{Z}_{[k]}$, and the rest $d-1$ indices for its columns.

Definition 4 [*Classical Mode-k Unfolding Matrix* (Kolda and Bader 2009)] Let $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \cdots \times n_d}$ be an dth-order tensor. The mode-k unfolding matrix of $\mathcal{Z}$ is denoted by $\mathbf{Z}_{(k)}$ of size $n_k \times \prod_{j \neq k}^{d} n_j$, whose elements obey

$$Z_{(k)}(j_k, \overline{j_1 \dots j_{k-1} j_{k+1} \dots j_d}) = Z(j_1, j_2, \dots, j_d). \tag{8}$$

where kth indices enumerate the rows of $\mathbf{Z}_{(k)}$, and the rest $d-1$ indices for its columns.

Definition 5 [*Tensor Subchains* (Zhao et al. 2016)]

- $\mathcal{G}^{\leq k} \in \mathbb{R}^{m_1 \times \prod_{j=1}^{k} n_j \times m_{k+1}}$ with lateral slice matrices $\mathbf{G}^{\leq k}(\overline{j_1 j_2 \dots j_k}) = \prod_{i=1}^{k} \mathbf{G}_i(j_i)$.
- $\mathcal{G}^{>k} \in \mathbb{R}^{m_{k+1} \times \prod_{j=k+1}^{d} n_j \times m_1}$ with lateral slice matrices $\mathbf{G}^{>k}(\overline{j_{k+1} j_{k+2} \dots j_d}) = \prod_{i=k+1}^{d} \mathbf{G}_i(j_i)$.
- $\mathcal{G}^{\neq k} \in \mathbb{R}^{m_{k+1} \times \prod_{j=1, j \neq k}^{d} n_j \times m_k}$ with slice matrices $\mathbf{G}^{\neq k}(\overline{j_{k+1} \dots j_d j_1 \dots j_{k-1}}) = \prod_{i=k+1}^{d} \mathbf{G}_i(j_i) \prod_{i=1}^{k-1} \mathbf{G}_i(j_i)$.

3 Proposed STF and TR-STF algorithms

In this section, we define STF, design its algorithm, and then apply it to TR decomposition, thus proposing the TR-STF algorithm.

3.1 Scaled tri-factorization (STF) algorithm

As illustrated in Sect. 2.2, the condition of FTF $m \ll \min(n_1, n_2)$ is not suitable for matrices without low-rankness. Based on this fact, we define the following factorization.

Definition 6 (*STF*) For a matrix $\mathbf{Z} \in \mathbb{R}^{n_1 \times n_2}$, STF of $\mathbf{Z}$ is

$$\mathbf{Z} = \mathbf{ABC} + \mathbf{N}, \tag{9}$$

where $\mathbf{N}$ denotes the error; $\mathbf{B} \in \mathbb{R}^{R \times R}$; $\mathbf{A} \in \mathbb{R}^{n_1 \times R}$ is column orthogonal, i.e., $\mathbf{A}^T\mathbf{A} = \mathbf{I}$, $\mathbf{C} \in \mathbb{R}^{R \times n_2}$ is row orthogonal, i.e., $\mathbf{CC}^T = \mathbf{I}$; and R denotes an upper bound on the rank of $\mathbf{Z}$ satisfying $\text{Rank}(Z) \leq R \leq \min\{n_1, n_2\}$.

With STF defined, we need to solve the following problem to find three matrices in Eq. (9) that satisfy $\|\mathbf{Z}-\mathbf{ABC}\|_F^2 \leq \delta$, where δ is a positive tolerance.

$$\arg\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \|\mathbf{Z}-\mathbf{ABC}\|_F^2,\ s.t.\ \mathbf{A}^T\mathbf{A}=\mathbf{I}, \mathbf{C}\mathbf{C}^T=\mathbf{I}. \tag{10}$$

Problem (10) is convex for each variable when the other two variables are fixed; therefore, we update $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ alternately. We initialize the three variables as $\mathbf{A}^0 = \text{eye}(n_1, R)$, $\mathbf{B}^0 = \text{eye}(R, R)$, and $\mathbf{C}^0 = \text{eye}(R, n_2)$, then the three variables are updated alternately.

1) $\mathbf{A}$-subproblem: With $\mathbf{B}$ and $\mathbf{C}$ are fixed, we update $\mathbf{A}$ by

$$\mathbf{A}^{t+1} = \arg\min_{\mathbf{A}^T\mathbf{A}=\mathbf{I}} \|\mathbf{Z}-\mathbf{A}\mathbf{B}^t\mathbf{C}^t\|_F^2, \tag{11}$$

which has a closed-form solution (Shen et al. 2014; Wen et al. 2012):

$$\mathbf{A}^{t+1} = \hat{\mathbf{A}}(:, 1:R), \tag{12}$$

where $\hat{\mathbf{A}} = \text{qr}(\mathbf{Z}(\mathbf{C}^t)^T)$.

2) $\mathbf{C}$-subproblem: With $\mathbf{A}$ and $\mathbf{B}$ are fixed, we update $\mathbf{C}$ by

$$\mathbf{C}^{t+1} = \arg\min_{\mathbf{C}\mathbf{C}^T=\mathbf{I}} \|\mathbf{Z}-\mathbf{A}^{t+1}\mathbf{B}^t\mathbf{C}\|_F^2, \tag{13}$$

which has a closed-form solution (Shen et al. 2014; Wen et al. 2012):

$$\mathbf{C}^{t+1} = \hat{\mathbf{C}}^T, \tag{14}$$

where $\hat{\mathbf{C}} = \bar{\mathbf{C}}(:, 1:R)$, $\bar{\mathbf{C}} = \text{qr}(\mathbf{Z}^T\mathbf{A}^{t+1})$.

3) $\mathbf{B}$-subproblem: With $\mathbf{A}$ and $\mathbf{C}$ are fixed, we update $\mathbf{B}$ by

$$\mathbf{B}^{t+1} = \arg\min_{\mathbf{B}} \|\mathbf{Z}-\mathbf{A}^{t+1}\mathbf{B}\mathbf{C}^{t+1}\|_F^2, \tag{15}$$

which has a closed-form solution:

$$\mathbf{B}^{t+1} = (\mathbf{A}^{t+1})^T\mathbf{Z}(\mathbf{C}^{t+1})^T. \tag{16}$$

We summarize the solving procedure of problem (10) in Algorithm 1, named STF, therein k_{mit} and δ are the maximum iteration (maxit) and stopping tolerance. In this paper, we empirically set k_{mit} and δ to 1 and 10^{-6}, respectively.

Algorithm 1 Scaled Tri-Factorization (STF) Algorithm

Input: $\mathbf{Z} \in \mathbb{R}^{n_1 \times n_2}$
Parameter: Maxit k_{mit}, δ, and R
Output: $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$
Initialization: $\mathbf{A}^0 = \text{eye}(n_1, R)$, $\mathbf{B}^0 = \text{eye}(R, R)$, $\mathbf{C}^0 = \text{eye}(R, n_2)$
1: Let $t = 0$
2: **while** not converged and $t < k_{mit}$ and $\|\mathbf{Z}-\mathbf{A}^t\mathbf{B}^t\mathbf{C}^t\|_F^2 > \delta$ **do**
3: Update $\mathbf{A}^{t+1}$ by Eq. (12)
4: Update $\mathbf{C}^{t+1}$ by Eq. (14)
5: Update $\mathbf{B}^{t+1}$ by Eq. (16)
6: Check the convergence criterion: $\frac{\|\mathbf{B}^{t+1}-\mathbf{B}^t\|_F^2}{\|\mathbf{B}^t\|_F^2} < 10^{-6}$.
7: $t \leftarrow t+1$
8: **end while**
9: **return** $\mathbf{A} = \mathbf{A}^t$, $\mathbf{B} = \mathbf{B}^t$, $\mathbf{C} = \mathbf{C}^t$

Table 1 Comparison of SVD and STF on three random matrices

Matrix size	10,000×2000		10,000×6000		10,000×10,000	
	RelCha	Time (s)	RelCha	Time (s)	RelCha	Time (s)
SVD	4.80e−15	2.52	7.00e−15	65.24	7.83e−15	264.70
STF	**1.03e−15**	**1.57**	**1.36e−15**	**11.84**	**1.64−15**	**31.65**

Best values are highlighted in bold

To illustrate the advantages of STF in data representation, we compare it with the well-known SVD algorithm. Specifically, we randomly generate three matrices with the size of $10{,}000 \times 2000$, $10{,}000 \times 6000$, and $10{,}000 \times 10{,}000$ from the normal distribution, and then represent them by SVD and STF, respectively. For STF, we set R as the minimum of two dimensions of the matrix. Table 1 shows the comparison of running time and accuracy for these two approaches; the best values are shown in bold font. Both algorithms perform well in accuracy, but the time consumption of STF is always less than that of SVD, and the time advantage of STF becomes more obvious as the size increases. The reason for the shorter running time of STF compared to SVD is primarily attributed to the computational efficiency of the QR decomposition employed in STF. The QR decomposition is known to be computationally much cheaper than the SVD (The Singular Value Decomposition (SVD) 2002; Wen et al. 2012).

Motivation 2 Inspired by the powerful data representation ability and high efficiency of STF, we apply it to complex tensor networks.

3.2 TR-STF algorithm

In this section, we propose an algorithm, TR-STF, for fast and accurate TR decomposition.

Theorem 1 *Suppose tensor $\mathcal{Z}$ can be expressed by a TR decomposition. If the size of and the rank of k-unfolding matrix $\mathbf{Z}_{\langle k\rangle}$ are $\prod_{j=1}^{k} n_j \times \prod_{j=k+1}^{d} n_j$ and M_{k+1}, respectively, then there exists a TR decomposition with TR rank $\mathbf{m} = [m_1, m_2, \ldots, m_d]^T$ which has that $\exists\, k$, $M_{k+1} \le m_1 m_{k+1} \le \min\{\prod_{j=1}^{k} n_j, \prod_{j=k+1}^{d} n_j\}$, the equal sign holds if $\mathbf{Z}_{\langle k\rangle}$ is a full rank matrix.*

Proof Based on Eqs. (3) and (6), we obtain the following equation:

$$Z_{\langle k\rangle}(\overline{j_1 \ldots j_k}, \overline{j_{k+1} \ldots j_d}) = \mathrm{Tr}\left\{\prod_{i=1}^{d} \mathbf{G}_i(j_i)\right\}. \tag{17}$$

Because equation $\mathrm{Tr}\{\prod_{i=1}^{d} \mathbf{G}_i(j_i)\} = \langle \mathrm{vec}(\prod_{i=1}^{k} \mathbf{G}_i(j_i)), \mathrm{vec}(\prod_{i=k+1}^{d} \mathbf{G}_i^T(j_i))\rangle$ holds (Zhao et al. 2016), we obtain the following equation:

$$Z_{\langle k\rangle}(\overline{j_1 \ldots j_k}, \overline{j_{k+1} \ldots j_d}) = \left\langle \mathrm{vec}\left(\prod_{i=1}^{k} \mathbf{G}_i(j_i)\right), \mathrm{vec}\left(\prod_{i=k+1}^{d} \mathbf{G}_i^T(j_i)\right)\right\rangle. \tag{18}$$

According to the definition of tensor subchains, we rewrite the Eq. (18) as

$$Z_{\langle k\rangle}(\overline{j_1 \ldots j_k}, \overline{j_{k+1} \ldots j_d}) = \sum_{\beta_1, \beta_{k+1}} \mathbf{G}^{\le k}(\overline{j_1 \ldots j_k}, \overline{\beta_1 \beta_{k+1}}) \mathbf{G}^{>k}(\overline{\beta_1 \beta_{k+1}}, \overline{j_{k+1} \ldots j_d}). \tag{19}$$

That is,

$$\mathbf{Z}_{\langle k\rangle} = \mathbf{G}_{(2)}^{\leq k}(\mathbf{G}_{[2]}^{>k})^T, \tag{20}$$

where the classical mode-2 unfolding matrix $\mathbf{G}_{(2)}^{\leq k} \in \mathbb{R}^{\prod_{j=1}^{k} n_j \times m_1 m_{k+1}}$ and mode-2 unfolding matrix $\mathbf{G}_{[2]}^{>k} \in \mathbb{R}^{\prod_{j=k+1}^{d} n_j \times m_1 m_{k+1}}$. Since the rank of matrix $\mathbf{Z}\langle k\rangle$ is M_{k+1}, we can establish the following inequalities: $M_{k+1} \leq m_1 m_{k+1}$ and $M_{k+1} \leq \min\{\prod_{j=1}^{k} n_j, \prod_{j=k+1}^{d} n_j\}$. We can always find a value for $m_1 m_{k+1}$ that satisfies $m_1 m_{k+1} \leq \min\{\prod_{j=1}^{k} n_j, \prod_{j=k+1}^{d} n_j\}$. Therefore, we have $M_{k+1} \leq m_1 m_{k+1} \leq \min\{\prod_{j=1}^{k} n_j, \prod_{j=k+1}^{d} n_j\}$. If matrix $\mathbf{Z}\langle k\rangle$ is a full rank matrix, it means that $M_{k+1} = \min\{\prod_{j=1}^{k} n_j, \prod_{j=k+1}^{d} n_j\}$. Hence, the equality $M_{k+1} = m_1 m_{k+1} = \min\{\prod_{j=1}^{k} n_j, \prod_{j=k+1}^{d} n_j\}$ holds. □

For TR-STF algorithm, we usually choose a specific mode as the starting point (e.g., the first mode). According to Eqs. (18) and (19), TR decomposition can be easily written as

$$Z_{\langle 1\rangle}(j_1, \overline{j_2 \dots j_d}) = \sum_{\beta_1, \beta_2} \mathbf{G}^{\leq 1}(j_1, \overline{\beta_1 \beta_2})\mathbf{G}^{>1}(\overline{\beta_1 \beta_2}, \overline{j_2 \dots j_d}). \tag{21}$$

Subsequently, we represent $\mathbf{Z}_{\langle 1\rangle}$ by STF, i.e., $\mathbf{Z}_{\langle 1\rangle} = \mathbf{A}_1\mathbf{B}_1\mathbf{C}_1 + \mathbf{N}_1$, where $\mathbf{N}_1$ denotes the error and $R = m_1 m_2$. The first core $\mathcal{G}_1 \in \mathbb{R}^{m_1 \times n_1 \times m_2}$ can be estimated by $\mathbf{A}_1$ with correct reshaping and permutation. The subchain $\mathcal{G}^{>1} \in \mathbb{R}^{m_2 \times \prod_{j=2}^{d} n_j \times m_1}$ can be generated by correct reshaping and permutation of $\mathbf{B}_1\mathbf{C}_1$, which corresponds to the remaining $d-1$ dimensions of $\mathcal{Z}$. Then we reshape $\mathcal{G}^{>1}$ as a matrix $\mathbf{G}^{>1} \in \mathbb{R}^{m_2 n_2 \times \prod_{j=3}^{d} n_j m_1}$ whose elements can be expressed as

$$G^{>1}(\overline{\beta_2 j_2}, \overline{j_3 \dots j_d \beta_1}) = \sum_{\beta_3} \mathbf{G}_2(\overline{\beta_2 j_2}, \beta_3)\mathbf{G}^{>2}(\beta_3, \overline{j_3 \dots j_d \beta_1}). \tag{22}$$

Using STF, i.e., $\mathbf{G}^{>1} = \mathbf{A}_2\mathbf{B}_2\mathbf{C}_2 + \mathbf{N}_2$, where $\mathbf{N}_2$ denotes the error and $R = m_3$, we generate the second core $\mathcal{G}_2 \in \mathbb{R}^{m_2 \times n_2 \times m_3}$ by reshaping $\mathbf{A}_2$ and obtain $\mathcal{G}^{>2} \in \mathbb{R}^{m_3 \times \prod_{j=3}^{d} n_j \times m_1}$ by reshaping $\mathbf{B}_2\mathbf{C}_2$. We perform this procedure sequentially to obtain all d cores $\mathcal{G}_j \in \mathbb{R}^{m_j \times n_j \times m_{j+1}}$, $j = 1, 2, \dots, d$. We summarize the proposed TR-STF in Algorithm 2, where "reshape" and "permute" are Matlab commands and $\text{Rank}(\mathbf{G}^{>k-1}) \leq m_{k+1} \leq \min\{m_k n_k, \prod_{j=k+1}^{d} n_j m_1\}$, $k = 2, 3, \dots, d-1$ are the conditions in STF. It is worth noting that the decomposition will be different if we choose a different mode as the start point. Especially, we draw a schematic diagram of TR-STF for 4th-order tensor $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$, as shown in Fig. 4, in which we choose the first mode as the start point.

3.3 Complexity analysis

Complexity analysis of STF algorithm According to Algorithm 1, the complexity of update $\mathbf{A}$, $\mathbf{C}$, and $\mathbf{B}$ in each iteration is $\mathcal{O}(n_1 R^2 + n_1 n_2 R)$, $\mathcal{O}(n_2 R^2 + n_1 n_2 R)$, and $\mathcal{O}(n_1 n_2 R + n_2 R^2)$. Thus, the total complexity of STF is $\mathcal{O}(n_1 R^2 + 2n_2 R^2 + 3n_1 n_2 R)$.

Complexity analysis of TR-STF algorithm As shown in Algorithm 2, steps 2 and 7 have high complexity. Specifically, the complexity of step 2 and step 7 is $\mathcal{O}(P_1 R_1^2 + 2Q_1 R_1^2 + 3P_1 Q_1 R_1)$ and $\mathcal{O}(P_2 R_2^2 + 2Q_2 R_2^2 + 3P_2 Q_2 R_2)$, respectively, where $P_1 = n_1$, $Q_1 = \prod_{j=2}^{d} n_j$, $R_1 = m_1 m_2$, $P_2 = m_k n_k$, $Q_2 = \prod_{j=k+1}^{d} n_j m_1$, and $R_2 = m_{k+1}$. Thus, the total complexity of TR-STF is $\mathcal{O}(P_1 R_1^2 + 2Q_1 R_1^2 + 3P_1 Q_1 R_1 + \sum_{k=2}^{d-1}(P_2 R_2^2 + 2Q_2 R_2^2 + 3P_2 Q_2 R_2))$.

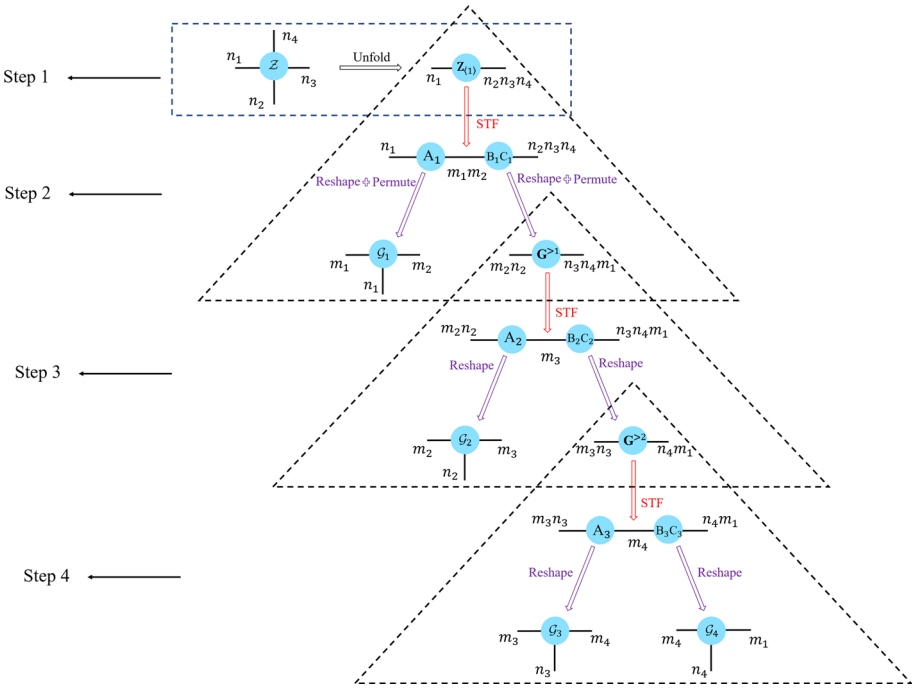

Fig. 4 TR-STF applied to a 4th-order tensor $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$. Step 1: Unfold the target tensor $\mathcal{Z}$ along the first mode to generate $\mathbf{Z}_{\langle 1 \rangle}$; Step 2: Represent $\mathbf{Z}_{\langle 1 \rangle}$ by STF (i.e., $\mathbf{Z}_{\langle 1 \rangle} = \mathbf{A}_1\mathbf{B}_1\mathbf{C}_1$) with $R = m_1m_2$, then obtain the first core $\mathcal{G}_1$ by $\mathbf{A}_1$ with correct reshaping and permutation and generate $\mathbf{G}^{>1}$ by $\mathbf{B}_1\mathbf{C}_1$ with correct reshaping and permutation; Step 3: Represent $\mathbf{G}^{>1}$ by STF (i.e., $\mathbf{G}^{>1} = \mathbf{A}_2\mathbf{B}_2\mathbf{C}_2$) with $R = m_3$, then obtain the second core $\mathcal{G}_2$ by reshaping $\mathbf{A}_2$ and generate $\mathbf{G}^{>2}$ by reshaping $\mathbf{B}_2\mathbf{C}_2$; Step 4: Represent $\mathbf{G}^{>2}$ by STF (i.e., $\mathbf{G}^{>2} = \mathbf{A}_3\mathbf{B}_3\mathbf{C}_3$) with $R = m_4$, then obtain the third core $\mathcal{G}_3$ by reshaping $\mathbf{A}_3$ and the fourth core $\mathcal{G}_4$ by reshaping $\mathbf{B}_3\mathbf{C}_3$

Algorithm 2 TR-STF Algorithm for TR Decomposition

Input: An dth-order tensor $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ and the predefined TR rank $\mathbf{m}$ satisfying $\text{Rank}(\mathbf{Z}_{\langle 1 \rangle}) \leq m_1m_2 \leq \min\{n_1, \prod_{j=2}^{d} n_j\}$ and $\text{Rank}(\mathbf{G}^{>k-1}) \leq m_{k+1} \leq \min\{m_kn_k, \prod_{j=k+1}^{d} n_jm_1\}, k = 2, 3, \ldots, d-1$
Output: Core tensors $\mathcal{G}_j$, $j = 1, 2, \ldots, d$

1: Select the first mode as the start point and generate $\mathbf{Z}_{\langle 1 \rangle}$
2: Represent $\mathbf{Z}_{\langle 1 \rangle}$ by STF: $\mathbf{Z}_{\langle 1 \rangle} = \mathbf{A}_1\mathbf{B}_1\mathbf{C}_1 + \mathbf{N}_1$, therein $R = m_1m_2$
3: $\mathcal{G}_1 \leftarrow \text{permute}(\text{reshape}(\mathbf{A}_1, [n_1, m_1, m_2]), [2, 1, 3])$
4: $\mathcal{G}^{>1} \leftarrow \text{permute}(\text{reshape}(\mathbf{B}_1\mathbf{C}_1, [m_1, m_2, \prod_{j=2}^{d} n_j]), [2, 3, 1])$
5: **for** $k = 2 : d - 1$ **do**
6: $\quad \mathbf{G}^{>k-1} \leftarrow \text{reshape}(\mathcal{G}^{>k-1}, [m_kn_k, \prod_{j=k+1}^{d} n_jm_1])$
7: $\quad$ Represent $\mathbf{G}^{>k-1}$ by STF: $\mathbf{G}^{>k-1} = \mathbf{A}_k\mathbf{B}_k\mathbf{C}_k + \mathbf{N}_k$, therein $R = m_{k+1}$
8: $\quad \mathcal{G}_k \leftarrow \text{reshape}(\mathbf{A}_k, [m_k, n_k, m_{k+1}])$
9: $\quad \mathcal{G}^{>k} \leftarrow \text{reshape}(\mathbf{B}_k\mathbf{C}_k, [m_{k+1}, \prod_{j=k+1}^{d} n_j, m_1])$
10: **end for**
11: $\mathcal{G}_d = \mathcal{G}^{>d-1}$

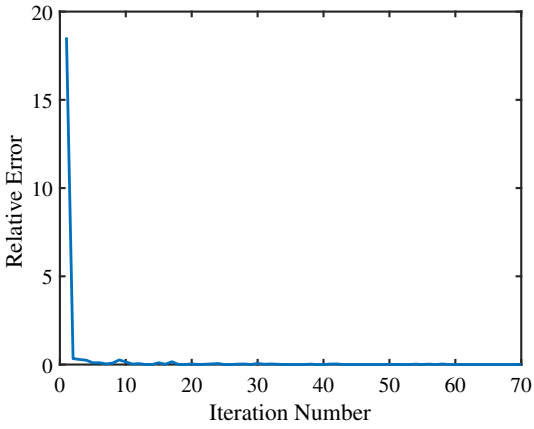

Fig. 5 Relative error curve of Algorithm 1

3.4 Convergence analysis

In this section, we propose two algorithms, i.e., Algorithms 1 and 2, where Algorithm 2 is a non-iterative algorithm that achieves convergence without the need for iterative processes (Zhao et al. 2016). Therefore, we only analyze the convergence of Algorithm 1. Specifically, we conduct an empirical analysis to evaluate the convergence of Algorithm 1. Figure 5 illustrates the convergence behavior of Algorithm 1 applied to the first band of the image (*x0023*) with a size of $321 \times 481 \times 3$ from the CBSD68 dataset. The figure clearly demonstrates that Algorithm 1 exhibits rapid and substantial decrease in the relative error, indicating efficient convergence.

4 Experiments

Because there is little work that focuses on TR decomposition, we compare the proposed TR-STF with four classical algorithms proposed by Zhao et al. (2016): TR-ALS, TR-ALSAR, TR-BALS, and TR-SVD, where the first approach needs to be predefined rank and the last three methods can estimate rank adaptively. In all the experiments, we perform all algorithms on MATLAB R2020a on an Intel(R) Core(TM) i9-10900KF CPU @ 3.70GHz 64.00GB RAM platform.

4.1 Highly oscillatory function

In this section, we test the performance of all five algorithms on three highly oscillatory functions, i.e., $f_1(x) = (x+1)sin(100(x+1)^2)$, $f_2(x) = x^{-\frac{1}{4}}sin(\frac{2}{3}x^{\frac{3}{2}})$, and $f_3(x) = \sin(x)\cos(x)$, as shown in Fig. 6. We evaluate the first two functions at n^d point, where n and d are considered to be the dimension and order of the tensor, respectively. We set n^d as 200^3, 100^4, 50^5, and 30^6, respectively, then reshape them into tensors with the size of $200\times200\times200$, $100\times100\times100\times100$, $50\times50\times50\times50\times50$, and $30\times30\times30\times30\times30\times30$, respectively. We set the RE and the maximum iteration steps as 10^{-4} and $20d$, respectively, and use the rank obtained from TR-SVD as the setting of TR-ALS. Table 2 shows the running time and RelCha comparison with different orders and dimensions; the best and secondary

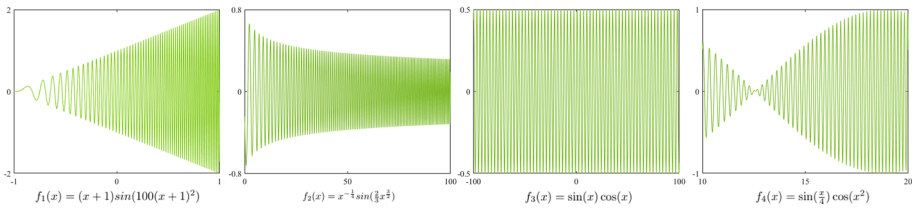

Fig. 6 Four highly oscillatory functions, where $f_1(x) = (x+1)\sin(100(x+1)^2)$, $f_2(x) = x^{-\frac{1}{4}}\sin(\frac{2}{3}x^{\frac{3}{2}})$, $f_3(x) = \sin(x)\cos(x)$, and $f_4(x) = \sin(\frac{x}{4})\cos(x^2)$

Table 2 Comparison of the performance of all five algorithms on two highly oscillatory functions (i.e., $f_1(x)$ and $f_2(x)$) at different points n^d

Algorithms	RelCha	Time (s)	RelCha	Time (s)
	$f_1(x), d = 3, n = 200$		$f_2(x), d = 3, n = 200$	
TR-ALS	2.91e−05	1.52	1.14e−05	0.78
TR-ALSAR	4.60e−05	8.90	5.15e−05	3.47
TR-BALS	**3.82e-06**	23.68	6.27e−06	31.12
TR-SVD	2.91e−05	0.21	1.14e−05	0.21
TR-STF (Ours)	8.45e−06	**0.03**	**3.42e-06**	**0.03**
	$d = 4, n = 100$		$d = 4, n = 100$	
TR-ALS	**3.13e-05**	12.26	1.82e−05	13.00
TR-ALSAR	1.80e−02	90.63	1.92e−02	89.50
TR-BALS	7.14e−05	9.77	**1.81e-05**	11.31
TR-SVD	**3.13e-05**	2.68	1.82e−05	3.12
TR-STF (Ours)	8.32e−05	**0.13**	7.28e−05	**0.12**
	$d = 5, n = 50$		$d = 5, n = 100$	
TR-ALS	5.52e−05	55.06	4.61e−05	59.77
TR-ALSAR	1.41e−02	405.91	1.55e−02	376.34
TR-BALS	5.24e−05	59.33	**3.49e-05**	60.38
TR-SVD	5.51e−05	8.35	4.61e−05	9.36
TR-STF (Ours)	**3.25e-05**	**0.60**	9.90e−05	**0.39**
	$d = 6, n = 30$		$d = 6, n = 30$	
TR-ALS	1.44e−05	365.64	**2.49e-05**	399.57
TR-ALSAR	1.96e−02	1409.52	5.32e−02	1271.90
TR-BALS	8.27e−06	261.39	5.44e−05	213.68
TR-SVD	1.44e−05	31.10	**2.49e-05**	34.53
TR-STF (Ours)	**3.82e-06**	**1.99**	9.32e−05	**0.89**

Best values are highlighted in bold and second best values are highlighted in underline

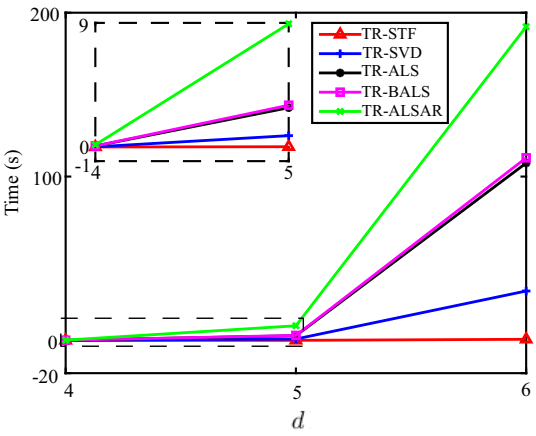

Fig. 7 The running time of TR decomposition of $f_3(x)$ at 30^d, $d = 4, 5, 6$, points, using the proposed TR-STF and other four state-of-the-art algorithms

values are bold and underlined, respectively. We find that all the algorithms leaving out TR-ALSAR can achieve good accuracy. The three ALS-based algorithms exhibit slower performance primarily because of their iterative nature, which involves repeated iterations during the computation. In contrast, the non-iterative algorithms, TR-STF and TR-SVD, demonstrate faster execution times. Among them, TR-STF achieves the best computational speed due to the utilization of the computationally more efficient QR decomposition. For example, when we decompose $f_2(x)$ at $d = 6$ and $n = 30$, the speed of the proposed TR-STF is improved as much as 448× higher than TR-ALS, 1429× higher than TR-ALSAR, 240× higher than TR-BALS, 38× higher than TR-SVD. Moreover, to more intuitively highlight the high efficiency of the proposed TR-STF, we evaluate $f_3(x)$ at n^d, $n = 30$, $d = 4, 5, 6$, points (when $d > 6$, the computer used for this work does not have enough memory to run these algorithms). Specifically, we reshape these values into 4th-order, 5th-order, and 6th-order tensors, respectively, then decompose each of them using five algorithms. We record the time required for five algorithms to achieve similar accuracy. The results are shown in Fig. 7. It can be seen that the proposed TR-STF achieves an absolute advantage in time cost.

4.2 Randomly simulated tensor data

In this experiment, we consider randomly generated tensors with fixed rank and variable dimension. Specifically, we randomly generate four cores $\mathcal{G}_i \in \mathbb{R}^{5\times n\times 5}$, $i = 1, 2, 3, 4$, where dimension n ranges from 20 to 100. Then, we use $\mathcal{G}_i$ to produce 4th-order tensor $\mathcal{Z}$ whose rank is $[5, 5, 5, 5]^T$. Subsequently, we apply the five algorithms to $\mathcal{Z}$; the RE for TR-ALS, TR-ALSAR, and TR-SVD is set as 10^{-4}, and the maxit for three ALS-based algorithms is set as $10d$ (where d is the order of the tensor). Table 3 and Fig. 8 report the accuracy and time results, respectively; the best and secondary values of RelCha in Table 3 are bold and underlined, respectively. We observe that TR-ALSAR exhibits poor accuracy performance, which can be attributed to the fact that its estimated TR rank is too small to capture the full complexity of the input data, resulting in reduced accuracy compared to other algorithms. In addition, the running time of TR-ALSAR increases rapidly with the increase of n. TR-ALS and TR-BALS have similar running times and accuracy performance, and TR-BALS can accurately estimate rank. The rank estimated by TR-SVD is large, but its performance is

Table 3 Comparison of all the algorithms on randomly simulated tensors with different dimensions n

Algorithms	n=20		n=40		n=60		n=80		n=100	
	RelCha	$\overline{m}$	RelCha	$\overline{m}$	RelCha	$\overline{m}$	RelCha	$\overline{m}$	RelCha	$\overline{m}$
TR-ALS	9.69e−05	–	7.39e−05	–	3.83e−05	–	5.58e−05	–	4.87e−05	–
TR-ALSAR	0.66	1	0.76	1	0.99	1	0.99	1	0.99	1
TR-BALS	5.45e−05	5	8.03e−05	5	1.99e−05	5	8.57e−05	5	4.83e−05	5
TR-SVD	4.27e−15	42.25	**3.46e-15**	65	**3.69e-15**	65	**3.57e-15**	65	**4.16e-15**	65
TR-STF (Ours)	**1.29e-15**	–	2.02e−13	–	3.71e−15	–	1.50e−14	–	5.70e−14	–

Best values are highlighted in bold and second best values are highlighted in underline
$\overline{m}$ denotes the average of TR rank

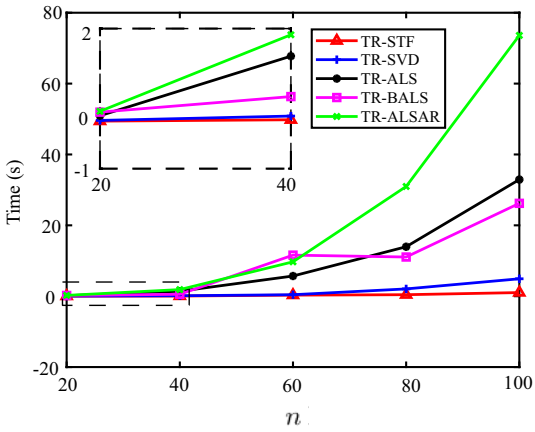

Fig. 8 Running time comparison of all the algorithms on randomly simulated tensor data with different dimensions n

Table 4 Comparison of all five algorithms on two real-world image data sets

Data set	Algorithms	RelCha	m_{max}	Time (s)
PU	TR-ALS	1.25e−14	127	184.81
	TR-ALSAR	0.17	10	0.26
	TR-BALS	5.39e−15	127	1.02
	TR-SVD	7.97e−15	512	0.78
	TR-STF (Ours)	**1.01e-15**	93	**0.02**
CBSD68	TR-ALS	–	–	–
	TR-ALSAR	0.38	2	19.02
	TR-BALS	0.37	3	6.38
	TR-SVD	8.13e−15	3264	16.91
	TR-STF (Ours)	**2.71e-15**	3300	**3.34**

Best values are highlighted in bold and second best values are highlighted in underline
$m_{\max}$ denotes the maximum value in TR rank

excellent. In general, for randomly simulated data, the proposed TR-STF yields secondary results in accuracy and more advantages in efficiency.

4.3 Real-world image data sets

In this section, we consider the performance of all algorithms on two real-world image data sets, the Pavia University (PU) Xu et al. (2020) and the CBSD68 data set. The maxit is set as $10d$ (where d is the order of the tensor) for three ALS-based algorithms in these two data sets.

PU data set The PU data set is a low-rank hyperspectral image with the size of $610 \times 340 \times 103$; we retain 93 bands by removing the low signal-to-noise ratio (SNR) bands, and then choose the up-left $64 \times 64 \times 93$ cube as the reference. We set the RE as 10^{-4} and use the rank obtained from TR-SVD as the setting of TR-ALS. The results are shown in Table 4; the best and secondary values of RelCha and running time are bold and underlined, respectively. We find that the performance of TR-ALSAR is unsatisfactory in terms of accuracy due to the small automatically estimated TR ranks. In contrast, TR-ALS, TR-BALS, TR-SVD and the

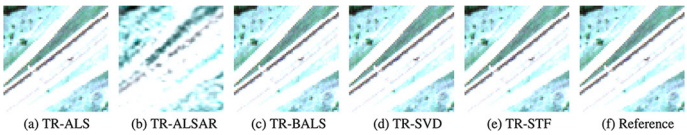

Fig. 9 Comparison of visual results on the PU data set. This figure shows images consisting of 10th, 20th, and 30th bands of results represented by all five algorithms and the reference image

proposed TR-STF can achieve the satisfactory accuracy. Moreover, the proposed algorithm also has more advantages in running time. For example, TR-STF is nearly 39 times faster than TR-SVD and 9240 times faster than TR-ALS.

CBSD68 data set: The CBSD68 data set contains 68 color images with size of $481 \times 321 \times 3$ and $321 \times 481 \times 3$. We first downsample them to $256 \times 256 \times 3$, then stack these images into a tensor $\mathcal{Z} \in \mathbb{R}^{256\times256\times3\times68}$. Here, $\mathcal{Z}$ is not low-rank since this data set describes 68 different scenarios. We perform four algorithms on tensor $\mathcal{Z}$ leaving out TR-ALS because its memory requirements outstrip the equipment used in these experiments. The RE is set as 10^{-5}, and the results are presented in Table 4; the best and secondary values of RelCha and running time are bold and underlined, respectively. We find that images recovered by TR-ALSAR and TR-BALS are not good, while TR-SVD and the proposed TR-STF can represent the tensor $\mathcal{Z}$ well; with respect to efficiency, TR-STF is more than five times faster than TR-SVD.

Moreover, we show the visual comparison of the results of the PU and the CBSD68 data sets, respectively. Figure 9 shows the images consisting of 10th, 20th, and 30th bands of results generated by all algorithms and of the reference image. We find that all algorithms except TR-ALSAR can represent the PU data set well. For the CBSD68 data set, the performance of TR-ALSAR and TR-BALS is unsatisfactory, while TR-SVD and TR-STF can perfectly express this data, as shown in detail in Fig. 10. In general, the performance of TR-ALSAR is unsatisfactory for these two real data sets. The reason may be that the estimation of rank is not accurate. The performance of TR-BALS is satisfactory for the PU data set, and the accuracy of TR-ALS is good, but it is not dominant in efficiency. The rank estimated by TR-SVD is usually large, but its efficiency and accuracy are the best of the classical algorithms. The proposed TR-STF has more advantages than TR-SVD.

4.4 Feature extraction for classification

Motivated by experiments in Zhao et al. (2016), we use all five algorithms for feature extraction for classification. For this experiment, we select the COIL-100 data set[4] using 7200 images with the size of $128 \times 128 \times 3$, which describe 100 different objects from 72 different directions. TR-SVD, TR-ALSAR, and TR-BALS also adopt a unified input rank. The RE and the maxit are set as 10^{-5} and $10d$ (where d is the order of the tensor), respectively. We first downsample each of the images to the size of $32 \times 32 \times 3$, then stack them into a tensor $\mathcal{Z} \in \mathbb{R}^{32\times32\times3\times7200}$. Next, we use all the algorithms on $\mathcal{Z}$ to estimate cores $\mathcal{G}_i \in \mathbb{R}^{m\times n_i\times m}$, $i = 1, 2, 3, 4$, with predefined rank ranging from 2 to 4, where the fourth core $\mathcal{G}_4$ can be used as the latent TR features while the subchain $\mathcal{G}^{\neq4}$ can be regarded as the basis of latent subspace. By making a permutation and reshaping, we generate a feature matrix $\mathbf{G}_4 \in \mathbb{R}^{7200\times m^2}$, then use the k-nearest neighbor algorithm[5] with $k = 1$ to classification. Table 5 presents the

[4] The details of this data set can be found at https://www.kaggle.com/jessicali9530/coil100.

[5] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.

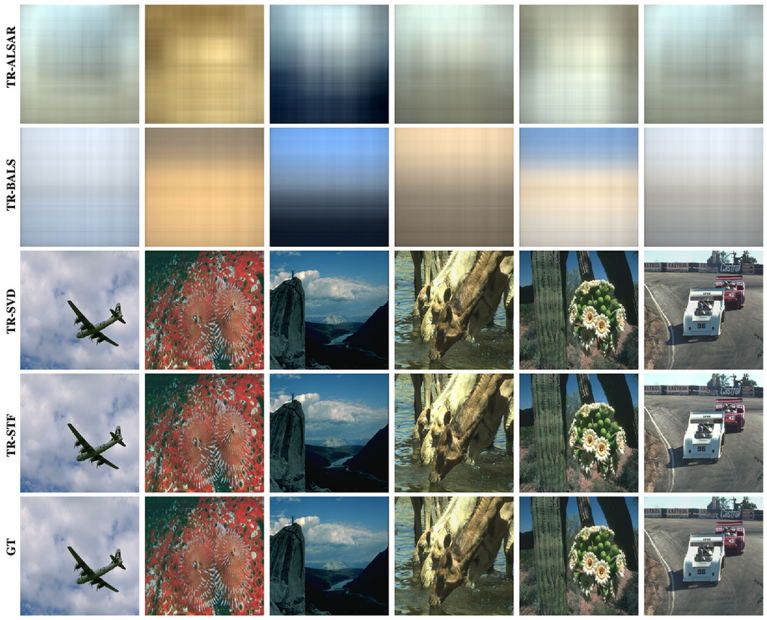

Fig. 10 Comparison of visual results on the CBSD68 data set. The 1st–4th row shows the results of TR-ALSAR, TR-BALS, TR-SVD, and TR-STF, the 5th row shows the corresponding ground truth (GT)

Table 5 Comparison of all the algorithms on the COIL-100 data set for classification with different TR ranks $[m, m, m, m]^T$

Algorithms	Acc (%)	Time (s)	Acc (%)	Time (s)
	$m = 2$		$m = 3$	
TR-ALS	92.56	21.09	98.10	42.19
TR-ALSAR	**92.69**	14.89	98.51	23.54
TR-BALS	79.71	5.96	86.53	8.73
TR-SVD	87.19	0.71	**99.25**	0.85
TR-STF (Ours)	88.79	**0.05**	95.17	**0.12**
	$m = 4$		$m = 5$	
TR-ALS	99.25	79.15	**99.81**	152.88
TR-ALSAR	99.21	42.86	99.57	77.44
TR-BALS	86.65	12.49	88.96	11.24
TR-SVD	**99.29**	0.99	99.56	1.19
TR-STF (Ours)	98.24	**0.28**	99.00	**0.57**

Best values are highlighted in bold and second best values are highlighted in underline

results; the best and secondary values of classification accuracy (Acc) and running time are bold and underlined, respectively. We find that the classification accuracy of each algorithm is higher with the increase in rank. In addition, the advantage of the proposed TR-STF in efficiency is still readily apparent.

5 Proposed TT-STF algorithm

In this section, we apply the sequential STFs to TT decomposition, thus proposing the TT-STF algorithm. Specifically, we briefly introduce the TT decomposition and then present the TT-STF algorithm.

5.1 Tensor train (TT) decomposition

This section introduces three forms of TT decomposition (Oseledets 2011).

For an dth-order tensor $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \cdots \times n_d}$, the TT decomposition in the element-wise form is represented by

$$Z(j_1, j_2, \ldots, j_d) = \mathbf{G}_1(j_1)\mathbf{G}_2(j_2)\ldots\mathbf{G}_d(j_d), \tag{23}$$

where $\mathbf{G}_i(j_i) \in \mathbb{R}^{m_i \times m_{i+1}}$ are the j_ith lateral slice of the ith core $\mathcal{G}_i \in \mathbb{R}^{m_i \times n_i \times m_{i+1}}$. The product of these parameter-dependent matrices is a matrix of size $m_1 \times m_{d+1}$, so "boundary conditions" $m_1 = m_{d+1} = 1$ have been imposed. Therefore, Eq. (23) can be rewritten as

$$Z(j_1, j_2, \ldots, j_d) = \mathbf{g}_1(j_1)\mathbf{G}_2(j_2)\ldots\mathbf{g}_d(j_d), \tag{24}$$

$\mathbf{g}_1(j_1)$ is the j_1th row vector of the first core $\mathbf{G}_1 \in \mathbb{R}^{n_1 \times m_2}$, $\mathbf{g}_d(j_d)$ is the j_dth column vector of the last core $\mathbf{G}_d \in \mathbb{R}^{m_d \times n_d}$. We rewrite (23) in the index form, i.e.,

$$Z(j_1, j_2, \ldots, j_d) = \sum_{\beta_1, \beta_2, \ldots, \beta_{d+1}} G_1(\beta_1, j_1, \beta_2)G_2(\beta_2, j_2, \beta_3)\ldots G_d(\beta_d, j_d, \beta_{d+1}), \tag{25}$$

where $\beta_{d+1} = \beta_1 = 1$, $1 \le \beta_k \le m_k$, $k \in \{2, \ldots, d\}$; $1 \le j_i \le n_i$, $i \in \{1, \ldots, d\}$; β_k is the index of latent dimensions; and j_i is the index of data dimensions. Based on Eq. (25), we can express TT decomposition in the tensor form, given by

$$\mathcal{Z} = \sum_{\beta_2, \ldots, \beta_d} \mathbf{g}_1(\beta_2) \circ \mathbf{g}_2(\beta_2, \beta_3) \circ \cdots \circ \mathbf{g}_d(\beta_d)^T, \tag{26}$$

where the symbol "$\circ$" denotes the outer product of vectors, $\mathbf{g}_1(\beta_2)$ and $\mathbf{g}_d(\beta_d)$ are β_2th row and β_dth column of $\mathbf{G}_1$ and $\mathbf{G}_d$, respectively, $\mathbf{g}_i(\beta_i, \beta_{i+1}) \in \mathbb{R}^{n_i}$ is the (β_i, β_{i+1})th mode-2 fiber of $\mathcal{G}_i$, therein $i \in \{2, 3, \ldots, d-1\}$.

5.2 TT-STF algorithm

In this section, we define the matrix subchain similar to tensor subchain for subsequent description, and then propose TT-STF algorithm.

Definition 7 (*Matrix Subchains*)

- $\mathbf{G}^{\le k} \in \mathbb{R}^{\prod_{j=1}^{k} n_j \times m_{k+1}}$ with row vector $\mathbf{g}^{\le k}(\overline{j_1 j_2 \ldots j_k}) = \mathbf{g}_1(j_1)\mathbf{G}_2(j_2)\ldots\mathbf{G}_k(j_k)$.
- $\mathbf{G}^{>k} \in \mathbb{R}^{m_{k+1} \times \prod_{j=k+1}^{d} n_j}$ with column vector $\mathbf{g}^{>k}(\overline{j_{k+1} j_{k+2} \ldots j_d}) = \mathbf{G}_{k+1}(j_{k+1})\mathbf{G}_{k+2}(j_{k+2})\ldots\mathbf{g}_d(j_d)$.

Next, we propose a fast and accurate algorithm, TT-STF, for TT decomposition using sequential STFs with theoretical proof.

Theorem 2 *Suppose tensor $\mathcal{Z}$ can be expressed by a TT decomposition. If the size of and the rank of k-unfolding matrix $\mathbf{Z}_{\langle k\rangle}$ are $\prod_{j=1}^{k} n_j \times \prod_{j=k+1}^{d} n_j$ and M_{k+1}, respectively, then there exists a TT decomposition with TT rank $\mathbf{m} = [m_2, m_3, \ldots, m_d]^T$ which has that $\exists$ k, $M_{k+1} \leq m_{k+1} \leq \min\{\prod_{j=1}^{k} n_j, \prod_{j=k+1}^{d} n_j\}$, the equal sign holds if $\mathbf{Z}_{\langle k\rangle}$ is a full rank matrix.*

Proof Based on Eqs. (6) and (24), we obtain the following equation:

$$Z_{\langle k\rangle}(\overline{j_1 \ldots j_k}, \overline{j_{k+1} \ldots j_d}) = \mathbf{g}_1(j_1)\mathbf{G}_2(j_2)\ldots\mathbf{g}_d(j_d). \tag{27}$$

According to the definition of matrix subchains, we rewrite Eq. (27) as

$$Z_{\langle k\rangle}(\overline{j_1 \ldots j_k}, \overline{j_{k+1} \ldots j_d}) = \sum_{\beta_{k+1}} \mathbf{G}^{\leq k}(\overline{j_1 \ldots j_k}, \beta_{k+1})\mathbf{G}^{>k}(\beta_{k+1}, \overline{j_{k+1} \ldots j_d}). \tag{28}$$

That is,

$$\mathbf{Z}_{\langle k\rangle} = \mathbf{G}^{\leq k}\mathbf{G}^{>k}, \tag{29}$$

where the matrix $\mathbf{G}^{\leq k} \in \mathbb{R}^{\prod_{j=1}^{k} n_j \times m_{k+1}}$ and matrix $\mathbf{G}^{>k} \in \mathbb{R}^{m_{k+1} \times \prod_{j=k+1}^{d} n_j}$. Since the rank of matrix $\mathbf{Z}\langle k\rangle$ is M_{k+1}, we can establish the following inequalities: $M_{k+1} \leq m_{k+1}$ and $M_{k+1} \leq \min\{\prod_{j=1}^{k} n_j, \prod_{j=k+1}^{d} n_j\}$. We can always find a value for m_{k+1} that satisfies $m_{k+1} \leq \min\{\prod_{j=1}^{k} n_j, \prod_{j=k+1}^{d} n_j\}$. Therefore, we have $M_{k+1} \leq m_{k+1} \leq \min\{\prod_{j=1}^{k} n_j, \prod_{j=k+1}^{d} n_j\}$. If matrix $\mathbf{Z}\langle k\rangle$ is a full rank matrix, it means that $M_{k+1} = \min\{\prod_{j=1}^{k} n_j, \prod_{j=k+1}^{d} n_j\}$. Hence, the equality $M_{k+1} = m_{k+1} = \min\{\prod_{j=1}^{k} n_j, \prod_{j=k+1}^{d} n_j\}$ holds. □

For the TT-STF algorithm, we usually choose a specific mode as the starting point (e.g., the first mode). According to Eqs. (27)–(28), TT decomposition can be easily written as

$$Z_{\langle 1\rangle}(j_1, \overline{j_2 \ldots j_d}) = \sum_{\beta_2} \mathbf{G}^{\leq 1}(j_1, \beta_2)\mathbf{G}^{>1}(\beta_2, \overline{j_2 \ldots j_d}). \tag{30}$$

Subsequently, we represent $\mathbf{Z}_{\langle 1\rangle}$ by STF, i.e., $\mathbf{Z}_{\langle 1\rangle} = \mathbf{A}_1\mathbf{B}_1\mathbf{C}_1 + \mathbf{N}_1$, where $\mathbf{N}_1$ denotes the error and $R = m_2$. The first core $\mathbf{G}_1 \in \mathbb{R}^{n_1 \times m_2}$ can be estimated by $\mathbf{A}_1$. The subchain $\mathbf{G}^{>1} \in \mathbb{R}^{m_2 \times \prod_{j=2}^{d} n_j}$ can be generated by $\mathbf{B}_1\mathbf{C}_1$, which corresponds to the remaining $d-1$ dimensions of $\mathcal{Z}$. Then we reshape $\mathbf{G}^{>1}$ as matrix $\widehat{\mathbf{Z}_2} \in \mathbb{R}^{m_2 n_2 \times \prod_{j=3}^{d} n_j}$ whose elements can be expressed as

$$\widehat{Z_2}(\overline{\beta_2 j_2}, \overline{j_3 \ldots j_d}) = \sum_{\beta_3} \mathbf{G}_2(\overline{\beta_2 j_2}, \beta_3)\mathbf{G}^{>2}(\beta_3, \overline{j_3 \ldots j_d}). \tag{31}$$

Using STF, i.e., $\widehat{\mathbf{Z}_2} = \mathbf{A}_2\mathbf{B}_2\mathbf{C}_2 + \mathbf{N}_2$, where $\mathbf{N}_2$ denotes the error and $R = m_3$, we generate the second core $\mathcal{G}_2 \in \mathbb{R}^{m_2 \times n_2 \times m_3}$ by reshaping $\mathbf{A}_2$ and obtain $\mathbf{G}^{>2} \in \mathbb{R}^{m_3 \times \prod_{j=3}^{d} n_j}$ by $\mathbf{B}_2\mathbf{C}_2$, which corresponds to the remaining $d-2$ dimensions of $\mathcal{Z}$. We perform this procedure sequentially to obtain all d cores. Finally, the proposed TT-STF is summarized in Algorithm 3, where "reshape" is Matlab commands and $\text{Rank}(\widehat{\mathbf{Z}_k}) \leq m_{k+1} \leq \min\{m_k n_k, \prod_{j=k+1}^{d} n_j\}$, $k = 2, 3, \ldots, d-1$ are the conditions in STF. It is worth noting that the decomposition will be different if we choose a different mode as the start point. Especially, we draw a schematic diagram of TT-STF for 4th-order tensor $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$, as shown in Fig. 11, in which we choose the first mode as the start point.

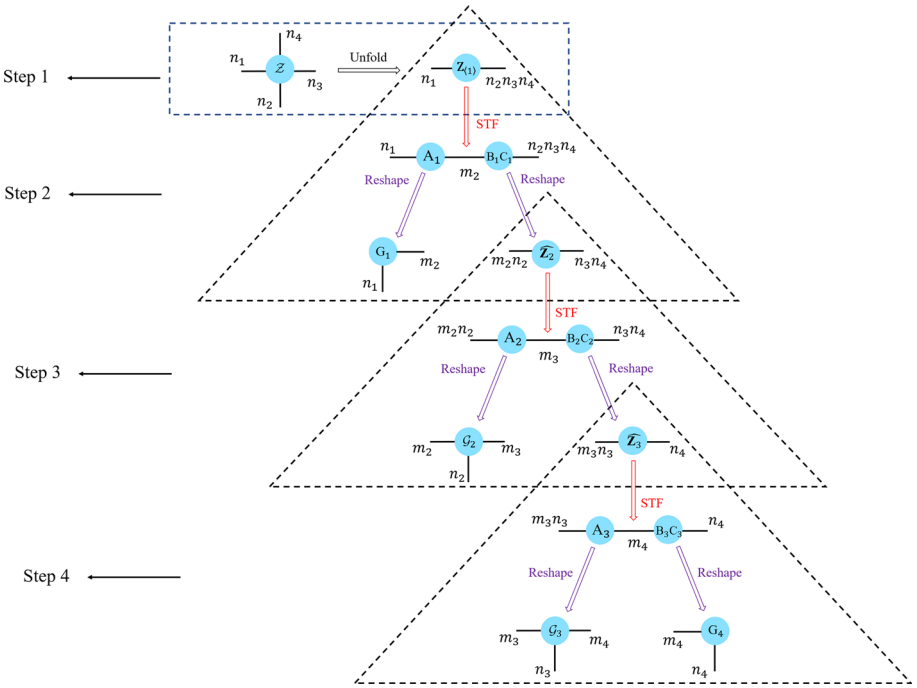

Fig. 11 TT-STF applied to a 4th-order tensor $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$. Step 1: Unfold the target tensor $\mathcal{Z}$ along the first mode to generate $\mathbf{Z}_{\langle 1 \rangle}$; Step 2: Represent $\mathbf{Z}_{\langle 1 \rangle}$ by STF (i.e., $\mathbf{Z}_{\langle 1 \rangle} = \mathbf{A}_1\mathbf{B}_1\mathbf{C}_1$) with $R = m_2$, then obtain the first core $\mathbf{G}_1$ by reshaping $\mathbf{A}_1$ and generate $\widehat{\mathbf{Z}_2}$ by reshaping $\mathbf{B}_1\mathbf{C}_1$; Step 3: Represent $\widehat{\mathbf{Z}_2}$ by STF (i.e., $\widehat{\mathbf{Z}_2} = \mathbf{A}_2\mathbf{B}_2\mathbf{C}_2$) with $R = m_3$, then obtain the second core $\mathcal{G}_2$ by reshaping $\mathbf{A}_2$ and generate $\widehat{\mathbf{Z}_3}$ by reshaping $\mathbf{B}_2\mathbf{C}_2$; Step 4: Represent $\widehat{\mathbf{Z}_3}$ by STF (i.e., $\widehat{\mathbf{Z}_3} = \mathbf{A}_3\mathbf{B}_3\mathbf{C}_3$) with $R = m_4$, then obtain the third core $\mathcal{G}_3$ by reshaping $\mathbf{A}_3$ and the fourth core $\mathbf{G}_4$ by reshaping $\mathbf{B}_3\mathbf{C}_3$

Algorithm 3 TT-STF Algorithm for TT Decomposition

Input: An dth-order tensor $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ and the predefined TT rank $\mathbf{m}$ satisfying $\text{Rank}(\mathbf{Z}_{\langle 1 \rangle}) \leq m_2 \leq \min\{n_1, \prod_{j=2}^{d} n_j\}$ and $\text{Rank}(\widehat{\mathbf{Z}_k}) \leq m_{k+1} \leq \min\{m_k n_k, \prod_{j=k+1}^{d} n_j\}$, $k = 2, 3, \ldots, d-1$

Output: Cores $\mathbf{G}_1 \in \mathbb{R}^{n_1 \times m_2}$, $\mathcal{G}_j \in \mathbb{R}^{m_i \times n_i \times m_{i+1}}$, $j = 2, 3, \ldots, d-1$, $\mathbf{G}_d \in \mathbb{R}^{m_d \times n_d}$, of TT decomposition

Select the first mode as the start point and obtain $\mathbf{Z}_{\langle 1 \rangle}$
for $k = 1 : d-1$ **do**
  if $k == 1$ **then**
    Represent $\mathbf{Z}_{\langle 1 \rangle}$ by STF: $\mathbf{Z}_{\langle 1 \rangle} = \mathbf{A}_1\mathbf{B}_1\mathbf{C}_1 + \mathbf{N}_k$, therein $R = m_2$
    $\mathbf{G}_1 \leftarrow \text{reshape}(\mathbf{A}_1, [n_1, m_2])$
  else
    $\widehat{\mathbf{Z}_k} \leftarrow \text{reshape}(\mathbf{G}^{>k-1}, [m_k n_k, \prod_{j=k+1}^{d} n_j])$
    Represent $\widehat{\mathbf{Z}_k}$ by STF: $\widehat{\mathbf{Z}_k} = \mathbf{A}_k\mathbf{B}_k\mathbf{C}_k + \mathbf{N}_k$, therein $R = m_{k+1}$
    $\mathcal{G}_k \leftarrow \text{reshape}(\mathbf{A}_k, [m_k, n_k, m_{k+1}])$
  end if
  $\mathbf{G}^{>k} \leftarrow \mathbf{B}_k\mathbf{C}_k$
end for
$\mathbf{G}_d = \mathbf{G}^{>d-1}$

Table 6 Comparison of TT-HSVD and TT-STF on highly oscillatory function $f_4(x)$ at different points n^d

Algorithms	RelCha	Time (s)	RelCha	Time (s)
	$d = 3, n = 200$		$d = 4, n = 100$	
TT-HSVD	9.48e−06	0.18	**1.66e-05**	1.98
TT-STF (Ours)	**1.97e-06**	**0.03**	3.15e−05	**0.09**
	$d = 5, n = 50$		$d = 6, n = 30$	
TT-HSVD	**1.84e-05**	5.61	3.79e−05	12.87
TT-STF (Ours)	2.45e−05	**0.36**	**4.33e-06**	**1.03**

Best values are highlighted in bold

5.3 Compare TT-STF with TT-HSVD

To evaluate the effectiveness of the proposed TT-STF, we compare TT-STF with the state-of-the-art TT-HSVD (Zniyed et al. 2020) on highly oscillatory function, $f_4(x) = \sin(\frac{x}{4})\cos(x^2)$ (see Fig. 6), to verify its effectiveness and efficiency. Specifically, we evaluate this function at n^d point, where n and d are considered to be the dimension and order of tensor, respectively. We set n^d as 200^3, 100^4, 50^5, and 30^6, then reshape them into tensors with the size of $200\times200\times200$, $100\times100\times100\times100$, $50\times50\times50\times50\times50$, and $30\times30\times30\times30\times30\times30$. The RE for TT-SVD is set as 10^{-4}, and the indices for TT-HSVD are set as 1, 2, 2, and 3, respectively. Table 6 shows the running time and RelCha comparison with different n and d; the best and secondary values of RelCha and running time are bold and underlined, respectively. We find that the proposed TT-STF has more advantages than TT-HSVD in efficiency under similar accuracy. For example, when $d = 6$ and $n = 30$, the running time of TT-STF is about one-twelfth that of TT-HSVD.

6 Conclusion

In this paper, we propose a stable TR-STF algorithm with theoretical proof based on sequential STFs to achieve fast and accurate TR decomposition. Specifically, to take advantage of the high efficiency of the existing FTF and overcome its limitation of only being efficient to low-rank matrices due to its strict condition, we define STF and design a corresponding algorithm. Compared with FTF, STF has a broader range of applications, can better preserve the latent information of data, and does not increase processing time appreciably. Equipped with this tool, we propose TR-STF for efficient TR decomposition. Extensive experimental results on multiple randomly simulated data, highly oscillatory functions, and real-world image data sets illustrate the high efficiency and effectiveness of the proposed TR-STF. As an extension, we use sequential STFs for fast and accurate TT decomposition with theoretical proof and propose TT-STF. Experimental results on highly oscillatory functions verify its effectiveness. The proposed TR-STF and TT-STF can generate excellent performance, but they cannot estimate rank adaptively, a topic for future research.

Acknowledgements The authors would like to thank the anonymous referees and editor for their valuable remarks, questions, and comments that enabled the authors to improve this paper. This research is supported by NSFC (12171072, 12271083), Natural Science Foundation of Sichuan Province (2022NSFSC0501), Key Projects of Applied Basic Research in Sichuan Province (Grant No. 2020YJ0216), and National Key Research and Development Program of China (Grant No. 2020YFA0714001).

Data Availability Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Declarations

Conflict of interest There is no conflict of interest.

References

Bozorgmanesh H, Hajarian M (2022) Triangular decomposition of CP factors of a third-order tensor with application to solving nonlinear systems of equations. J Sci Comput 90:74

Brachat J, Comon P, Mourrain B, Tsigaridas E (2010) Symmetric tensor decomposition. Linear Algebra Appl 433:1851–1872

Bro R (1997) PARAFAC. Tutorial and applications. Chem Intell Lab Syst 38(2):149–171

Cao X, Yao J, Xu Z, Meng D (2020) Hyperspectral image classification with convolutional neural network and active learning. IEEE Trans Geosci Remote Sens 58(7):4604–4616

Che M, Wei Y (2020) Multiplicative algorithms for symmetric nonnegative tensor factorizations and its applications. J Sci Comput 83:53

Che M, Wei Y, Yan H (2021) An efficient randomized algorithm for computing the approximate Tucker decomposition. J Sci Comput 88:32

Chen Y, Huang T-Z, He W, Yokoya N, Zhao X-L (2020) Hyperspectral image compressive sensing reconstruction using subspace-based nonlocal tensor ring decomposition. IEEE Trans Image Process 29:6813–6828

Cichocki A, Lee N, Oseledets A-H, Phan I, Zhao Q, Mandic DP (2016) Tensor networks for dimensionality reduction and large-scale optimization: part 1 low-rank tensor decompositions. Found Trends® Mach Learn 9:249–429

De Lathauwer L (2006) A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization. SIAM J Matrix Anal Appl 28(3):642–666

De Lathauwer L, De Moor B, Vandewalle J (2000) On the best rank-1 and rank-(R1,R2,. . .,RN) approximation of higher-order tensors. SIAM J Matrix Anal Appl 21(4):1324–1342

DE Silva V, Lim L-H (2008) Tensor rank and the ill-posedness of the best low-rank approximation problem. SIAM J Matrix Anal Appl 30:1084–1127

Dektor A, Rodgers A, Venturi D (2021) Rank-adaptive tensor methods for high-dimensional nonlinear PDEs. J Sci Comput 36:88

Deng L-J, Feng M, Tai X-C (2019) The fusion of panchromatic and multispectral remote sensing images via tensor-based sparse modeling and hyper-Laplacian prior. Inf Fusion 52:76–89

Deng L-J, Vivone G, Jin C, Chanussot J (2021) Detail injection-based deep convolutional neural networks for pansharpening. IEEE Trans Geosci Remote Sens 59(8):6995–7010

Deng L-J, Vivone G, Paoletti ME, Scarpa G, He J, Zhang Y, Chanussot J, Plaza A (2022) Machine learning in pansharpening: a benchmark, from shallow to deep networks. IEEE Geosci Remote Sens Mag 10(3):279–315

Deng S-Q, Deng L-J, Wu X, Ran R, Hong D, Vivone G (2023) PSRT: pyramid shuffle-and-reshuffle transformer for multispectral and hyperspectral image fusion. IEEE Trans Geosci Remote Sens 61:1–15. https://doi.org/10.1109/TGRS.2023.3244750

Dian R, Li S, Fang L (2019) Learning a low tensor-train rank representation for hyperspectral image super-resolution. IEEE Trans Neurral Netw Learn Syst 30(9):2672–2683

Ding M, Huang T-Z, Ji T-Y, Zhao X-L, Yang J-H (2019) Low-rank tensor completion using matrix factorization based on tensor train rank and total variation. J Sci Comput 81:941–964

Ding M, Huang T-Z, Zhao X-L, Ng MK, Ma T-H (2021) Tensor train rank minimization with nonlocal self-similarity for tensor completion. Inverse Probl Imaging 15(3):475–498

Fu X, Lin Z, Huang Y, Ding, X (2019) A variational pan-sharpening with local gradient constraints. In: Proceedings of IEEE conference on computer vision pattern recognition (CVPR), pp 10257–10266

Gnanasekaran DEA (2022) Hierarchical orthogonal factorization: sparse least squares problems. J Sci Comput 91:50

Goulart JHDM, Boizard M, Boyer R, Favier G, Comon P (2016) Tensor CP decomposition with structured factor matrices: algorithms and performance. IEEE J Sel Top Signal Process 10:757–769

Hashemizadeh M, Liu M, Miller J, Rabusseau G (2020) Adaptive tensor learning with tensor networks. In: Proceedings of NeurIPS 1st workshop on quantum tensor networks in machine learning

He W, Yokoya N, Yuan L-H, Zhao Q-B (2019) Remote sensing image reconstruction using tensor ring completion and total variation. IEEE Trans Geosci Remote Sens 57(11):8998–9009
He W, Yao Q, Chao L, Yokoya N, Zhao Q, Zhang H, Zhang L (2022) Non-Local Meets Global: an iterative paradigm for hyperspectral image restoration. IEEE Trans Pattern Anal Mach Intell 44(04):2089–2107
Hillar CJ, Lim L-H (2013) Most tensor problems are NP-hard. J ACM 60:1–39
Holtz S, Rohwedder T, Schneider R (2012) The alternating linear scheme for tensor optimization in the tensor train format. SIAM J Sci Comput 34(2):A683–A713
Huckle T, Waldherr K, Schulte-Herbriggen T (2013) Computations in quantum tensor networks. Linear Algebra Appl 438(2):750–781
Jiang J, Sanogo F, Navasca C (2022) Low-CP-rank tensor completion via practical regularization. J Sci Comput 91:18
Kolda TG, Bader BW (2009) Tensor decompositions and applications. SIAM Rev 51(3):455–500
Liu Y, Jiao LC, Shang F (2013) A fast tri-factorization method for low-rank matrix recovery and completion. Pattern Recogn 46(1):163–173
Luan Z, Ming Z, Wu Y (2023) Hankel tensor-based model and l1-tucker decomposition-based frequency recovery method for harmonic retrieval problem. Comput Appl Math 42:14
Orus R (2014) A practical introduction to tensor networks: matrix product states and projected entangled pair states. Ann Phys 349:117–158
Oseledets IV (2011) Tensor-train decomposition. SIAM J Sci Comput 33(5):2295–2317
Oseledets IV, Tyrtyshnikov EE (2009) Breaking the curse of dimensionality, or how to use SVD in many dimensions. SIAM J Sci Comput 31:3744–3759
Oseledets I, Tyrtyshnikov E (2010) TT-cross approximation for multidimensional arrays. Linear Algebra Appl 432(1):70–88
Qi L, Wang Q, Chen Y (2015) Three dimensional strongly symmetric circulant tensors. Linear Algebra Appl 482:207–220
Ran R, Deng L-J, Jiang T-X, Hu J-F, Chanussot J, Vivone G (2023) GuidedNet: a general CNN fusion framework via high-resolution guidance for hyperspectral image super-resolution. IEEE Trans Cybern. https://doi.org/10.1109/TCYB.2023.3238200
Shen Y, Wen Z, Zhang Y (2014) Augmented Lagrangian alternating direction method for matrix separation based on low-rank factorization. Optim Methods Softw 29(2):239–263
Sultonov A, Matveev S, Budzinskiy S (2023) Low-rank nonnegative tensor approximation via alternating projections and sketching. Comput Appl Math 42:68
Sun C-W, Huang T-Z, Xu T, Deng L-J (2023) NF-3DLogTNN: an effective hyperspectral and multispectral image fusion method based on nonlocal low-fibered-rank regularization. Appl Math Model 118:780–797
Tai X-C, Deng L-J, Yin K (2021) A multigrid algorithm for maxflow and Min-Cut problems with applications to multiphase image segmentation. J Sci Comput 101:87
The Singular Value Decomposition (SVD), chap. 4. Wiley, pp 261–288 (2002). https://doi.org/10.1002/0471249718.ch4. https://onlinelibrary.wiley.com/doi/abs/10.1002/0471249718.ch4
Tucker LR (1996) Some mathematical notes on three-mode factor analysis. Psychometrika 31(3):279–311
Wang Y, Yang Y (2022) Hot-svd: higher order t-singular value decomposition for tensors based on tensor-tensor product. Comput Appl Math 41:394
Wang H, Zhang F, Wang J, Huang T, Huang J, Liu X (2022) Generalized nonconvex approach for low-tubal-rank tensor recovery. IEEE Trans Neural Netw Learn Syst 33(8):3305–3319
Wang H, Peng J, Qin W, Wang J, Meng D (2023) Guaranteed tensor recovery fused low-rankness and smoothness. IEEE Trans Pattern Anal Mach Intell 1:1–17. https://doi.org/10.1109/TPAMI.2023.3259640
Wen Z, Yin W, Zhang Y (2012) Solving a low-rank factorization model for matrix completion by a non-linear successive over-relaxation algorithm. Math Program Comput 4:333–361
Xiao C, Yang C, Li M (2021) Efficient alternating least squares algorithms for low multilinear rank approximation of tensors. J Sci Comput 87:67
Xiao J-L, Huang T-Z, Deng L-J, Wu Z-C, Vivone G (2022) A new context-aware details injection fidelity with adaptive coefficients estimation for variational pansharpening. IEEE Trans Geosci Remote Sens 60:1–15
Xu C (2016) Hankel tensors, Vandermonde tensors and their positivities. Linear Algebra Appl 491:56–72
Xu T, Huang T-Z, Deng L-J, Zhao X-L, Huang J (2020) Hyperspectral image super-resolution using unidirectional total variation with Tucker decomposition. IEEE J Sel Top Appl Earth Obs Remote Sens 13:4381–4398
Xu T, Huang T-Z, Deng L-J, Yokoya N (2022) An iterative regularization method based on tensor subspace representation for hyperspectral image super-resolution. IEEE Trans Geosci Remote Sens 60:1–16. https://doi.org/10.1109/TGRS.2022.3176266
Xue J, Zhao Y, Liao W, Chan JC-W (2019) Nonlocal low-rank regularized tensor decomposition for hyperspectral image denoising. IEEE Trans Geosci Remote Sens 57(7):5174–5189

Xue J, Zhao Y, Liao W, Chan JC-W, Kong SG (2020) Enhanced sparsity prior model for low-rank tensor completion. IEEE Trans Neural Netw Learn Syst 31(11):4567–4581

Xue J, Zhao Y-Q, Bu Y, Liao W, Chan JC-W, Philips W (2021a) Spatial-spectral structured sparse low-rank representation for hyperspectral image super-resolution. IEEE Trans Image Process 30:3084–3097

Xue J, Zhao Y-Q, Huang S, Liao W, Chan JC-W, Kong SG (2021b) Multilayer sparsity-based tensor decomposition for low-rank tensor completion. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2021.3083931

Xue J, Zhao Y, Bu Y, Chan JC-W, Kong SG (2022) When Laplacian scale mixture meets three-layer transform: a parametric tensor sparsity for tensor completion. IEEE Trans Cybern 52(12):13887–13901. https://doi.org/10.1109/TCYB.2021.3140148

Zhao Q, Zhou G, Xie S, Zhang L, Cichocki A (2016) Tensor ring decomposition. arXiv:1606:05535

Zniyed Y, Boyer R, de Almeida ALF, Favier G (2020) A TT-based hierarchical framework for decomposing high-order tensors. SIAM J Sci Comput 42(2):822–848