SUPPLEMENTARY MATERIAL FOR 'ROBUST HIGH-ORDER TENSOR RECOVERY VIA NONCONVEX LOW-RANK APPROXIMATION'

Wenjin Qin[1], *Hailin Wang*[2], *Weijun Ma*[3], *Jianjun Wang*[1,*]

[1]School of Mathematics and Statistics, Southwest University, China
[2] School of Mathematics and Statistics, Xi'an Jiaotong University, China
[3] School of Information Engineering, Ningxia University, China

In this document, we present the algebraic framework associated with order-d ($d \geq 4$) tensor Singular Value Decomposition (t-SVD). To improve readability, the basic symbols and operators utilized for each section are summarized in Table 1, some of which originate from [1–6].

1. BASIC NOTIONS AND PRELIMINARIES

In this section, we introduce main notions and preliminaries concerning order-d tensor that are necessary for the whole paper.

Table 1: The main notions and preliminaries for order-d tensor.

Notations	Descriptions	Notations	Descriptions
$\lfloor t \rfloor$	nearest integer less than or equal to t	$\lceil t \rceil$	the one greater than or equal to t
$[d] = \{1, 2, \cdots, d-1, d\}$	set of the first d natural numbers	$\mathrm{A} \in \mathbb{C}^{n_1 \times n_2}$	matrix
A^*	conjugate transpose	$\mathrm{I}_n \in \mathbb{R}^{n \times n}$	$n \times n$ identity matrix
$\mathrm{tr}(\mathrm{A})$	matrix trace	$\langle \mathrm{A}, \mathrm{B} \rangle = \mathrm{tr}(\mathrm{A}^* \times \mathrm{B})$	matrix inner product
$\sigma_i(\mathrm{A})$	i-th singular value	$\mathrm{A} \otimes \mathrm{B}$	Kronecker product of A and B
$\Vert\mathrm{A}\Vert_q = (\sum_{i,j} \vert\mathrm{A}_{ij}\vert^q)^{\frac{1}{q}}$	matrix ℓ_q-norm	$\Vert\mathrm{A}\Vert_F = (\sum_{ij} \vert\mathrm{A}_{ij}\vert^2)^{1/2}$	matrix Frobenius norm
$\Vert\mathrm{A}\Vert_\infty = \max_{i,j} \vert\mathrm{A}_{ij}\vert$	matrix infinity norm	$\Vert\mathrm{A}\Vert = \max_i \sigma_i(\mathrm{A})$	matrix spectral norm
$\Vert\mathrm{A}\Vert_w = \sum_i w_i \sigma_i(\mathrm{A})$	matrix weighted nuclear norm	$\Vert\mathrm{A}\Vert_{w,\mathrm{S}_p} = \left(\sum_i w_i \left\vert\sigma_i(\mathrm{A})\right\vert^p\right)^{1/p}$	matrix weighted Schatten-p norm
$\Vert\mathrm{A}\Vert_\star = \sum_i \sigma_i(\mathrm{A})$	matrix nuclear norm	$\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$	order-d tensor
$(i_1, \cdots, i_d)$	tensor multiple indices	$\boldsymbol{\mathcal{A}}_{i_1 \cdots i_d}$ or $\boldsymbol{\mathcal{A}}(i_1, \cdots, i_d)$	$(i_1, \cdots, i_d)$-th entry
$\mathrm{A}_{(k)} \in \mathbb{R}^{n_k \times \prod_{j \neq k} n_j}$	mode-k unfolding of $\boldsymbol{\mathcal{A}}$	$\Vert\boldsymbol{\mathcal{A}}\Vert_\infty = \max_{i_1 \cdots i_d} \vert\boldsymbol{\mathcal{A}}_{i_1 \cdots i_d}\vert$	tensor infinity norm
$\Vert\boldsymbol{\mathcal{A}}\Vert_q = (\sum_{i_1 \cdots i_d} \vert\boldsymbol{\mathcal{A}}_{i_1 \cdots i_d}\vert^q)^{\frac{1}{q}}$	tensor ℓ_q-norm	$\boldsymbol{\mathcal{A}}(:, :, i_3, \cdots, i_d)$ or $\mathrm{A}^{(i_3, \cdots, i_d)}$	slice along mode-1 and mode-2
$\Vert\boldsymbol{\mathcal{A}}\Vert_F = (\sum_{i_1 \cdots i_d} \vert\boldsymbol{\mathcal{A}}_{i_1 \cdots i_d}\vert^2)^{1/2}$	tensor Frobenius norm	$\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \sum \langle \mathrm{A}^{(i_3, \cdots, i_d)}, \mathrm{B}^{(i_3, \cdots, i_d)} \rangle$	tensor inner product
$\boldsymbol{\mathcal{A}}_\mathrm{f} = \mathrm{fft}(\boldsymbol{\mathcal{A}}, [\,], i)$ for $i = 3, \cdots, d$	FFT along mode-3 to mode-d	$\boldsymbol{\mathcal{A}} = \mathrm{ifft}(\boldsymbol{\mathcal{A}}_\mathrm{f}, [\,], j)$ for $j = d, \cdots, 3$	inverse Fast Fourier Transform
$L(\boldsymbol{\mathcal{A}}) : \mathbb{C}^{n_1 \times \cdots \times n_d} \rightarrow \mathbb{C}^{n_1 \times \cdots \times n_d}$	generic invertible linear transform	$\boldsymbol{\mathcal{A}} *_L \boldsymbol{\mathcal{B}}$	linear transform L based t-product
$\boldsymbol{\mathcal{A}}_i \in \mathbb{R}^{n_1 \times \cdots \times n_{d-1}}$	order-$(d-1)$ tensor constructed by keeping the d-th index of $\boldsymbol{\mathcal{A}}$ fixed at i , $\boldsymbol{\mathcal{A}}_i := \boldsymbol{\mathcal{A}}(:, \cdots, :, i)$.		
$\mathrm{A}^j \in \mathbb{R}^{n_1 \times n_2}$	$\mathrm{A}^j = \boldsymbol{\mathcal{A}}(:, :, i_3, \cdots, i_d),\ j = (i_d - 1)n_3 \cdots n_{d-1} + \cdots + (i_4 - 1)n_3 + i_3,\ i_d \in \{1, \cdots, n_d\}$.		
$\boldsymbol{\mathcal{A}} \times_n \mathrm{U}$	the mode-n product of tensor $\boldsymbol{\mathcal{A}}$ with matrix U, $\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{A}} \times_n \mathrm{U} \Longleftrightarrow \mathrm{B}_{(n)} = \mathrm{U} \cdot \mathrm{A}_{(n)}$.		
$\boldsymbol{\mathcal{A}}_L \triangleq L(\boldsymbol{\mathcal{A}})$	$L(\boldsymbol{\mathcal{A}}) = \boldsymbol{\mathcal{A}} \times_3 \mathrm{U}_{n_3} \times_4 \mathrm{U}_{n_4} \cdots \times_d \mathrm{U}_{n_d}$, $\mathrm{U}_{n_i} \in \mathbb{C}^{n_i \times n_i}$ denotes an invertible transform matrix.		
$L^{-1}(\boldsymbol{\mathcal{A}})$	$L^{-1}(\boldsymbol{\mathcal{A}}) = \boldsymbol{\mathcal{A}} \times_d \mathrm{U}_{n_d}^{-1} \times_{d-1} \mathrm{U}_{n_{d-1}}^{-1} \cdots \times_3 \mathrm{U}_{n_3}^{-1}$, $L^{-1}(L(\boldsymbol{\mathcal{A}})) = \boldsymbol{\mathcal{A}}$.		
$\mathrm{circ}\,(\boldsymbol{\mathcal{A}})$	$\mathrm{circ}\,(\boldsymbol{\mathcal{A}}) = \begin{bmatrix} \boldsymbol{\mathcal{A}}_1 & \boldsymbol{\mathcal{A}}_{n_d} & \boldsymbol{\mathcal{A}}_{n_d-1} & \cdots & \boldsymbol{\mathcal{A}}_2 \\ \boldsymbol{\mathcal{A}}_2 & \boldsymbol{\mathcal{A}}_1 & \boldsymbol{\mathcal{A}}_{n_d} & \cdots & \boldsymbol{\mathcal{A}}_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\mathcal{A}}_{n_d} & \boldsymbol{\mathcal{A}}_{n_d-1} & \boldsymbol{\mathcal{A}}_{n_d-2} & \cdots & \boldsymbol{\mathcal{A}}_1 \end{bmatrix} \in \mathbb{R}^{n_1 n_d \times \cdots \times n_{d-2} n_d \times n_{d-1}}$.		
$\mathrm{bcirc}(\boldsymbol{\mathcal{A}})$	a $(n_1 n_3 \cdots n_d \times n_2 n_3 \cdots n_d)$ block circulant matrix at the base level of the operator $\mathrm{circ}(\boldsymbol{\mathcal{A}})$.		
$\mathrm{unfold}\,(\boldsymbol{\mathcal{A}})$	$\mathrm{unfold}\,(\boldsymbol{\mathcal{A}}) = [\boldsymbol{\mathcal{A}}_1, \boldsymbol{\mathcal{A}}_2, \cdots, \boldsymbol{\mathcal{A}}_{n_d-1}, \boldsymbol{\mathcal{A}}_{n_d}]^\mathrm{T} \in \mathbb{R}^{n_1 n_d \times n_2 \times \cdots \times n_{d-1}}$.		
$\mathrm{fold}(\boldsymbol{\mathcal{A}})$	the operation takes $\mathrm{unfold}\,(\boldsymbol{\mathcal{A}})$ back to order-d tensor form, i.e., $\mathrm{fold}(\mathrm{unfold}\,(\boldsymbol{\mathcal{A}}), n_d) = \boldsymbol{\mathcal{A}}$.		
$\mathrm{bunfold}(\boldsymbol{\mathcal{A}})$	a $(n_1 n_3 \cdots n_d \times n_2)$ matrix formed by applying $\mathrm{unfold}(\cdot)$ repeatedly until a block matrix result.		
$\mathrm{bfold}(\boldsymbol{\mathcal{A}})$	the operation takes $\mathrm{bunfold}\,(\boldsymbol{\mathcal{A}})$ back to order-d tensor form, i.e., $\mathrm{bfold}(\mathrm{bunfold}\,(\boldsymbol{\mathcal{A}})) = \boldsymbol{\mathcal{A}}$.		
$\mathrm{bdiag}(\boldsymbol{\mathcal{A}})$	$\mathrm{bdiag}(\boldsymbol{\mathcal{A}}) = \mathrm{diag}(\mathrm{A}^1, \mathrm{A}^2, \cdots, \mathrm{A}^{J-1}, \mathrm{A}^J) \in \mathbb{R}^{n_1 n_3 \cdots n_d \times n_2 n_3 n_d},\ J = n_3 \cdots n_d$.		
$\boldsymbol{\mathcal{A}} \triangle \boldsymbol{\mathcal{B}}$	face-wise product of two order-d tensor, $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \triangle \boldsymbol{\mathcal{B}} \Longleftrightarrow \mathrm{bdiag}(\boldsymbol{\mathcal{C}}) = \mathrm{bdiag}(\boldsymbol{\mathcal{A}}) \cdot \mathrm{bdiag}(\boldsymbol{\mathcal{B}})$.		

*✉ wjj@swu.edu.cn.

Algorithm 1: Generic order-d t-product [6], tpro-L($\mathcal{A}, \mathcal{B}, L$).

Input: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times \cdots \times n_d}$,$\mathcal{B} \in \mathbb{R}^{n_2 \times l \times n_3 \times \cdots \times n_d}$,
and corresponding matrices $\{\mathrm{U}_{n_i}\}_{i=3}^{d}$ of linear transforms L.
Output: $\mathcal{C} = \mathcal{A} *_L \mathcal{B} \in \mathbb{R}^{n_1 \times l \times n_3 \times \cdots \times n_d}$.

Compute the result of linear transform on $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{A}_L \leftarrow L(\mathcal{A})$, $\mathcal{B}_L \leftarrow L(\mathcal{B})$;
Compute each matrix slice of $\mathcal{C}_L$ by
for $i_3 \in \{1, \cdots, n_3\}, \cdots, i_d \in \{1, \cdots, n_d\}$ **do**
  $\mathcal{C}_L(:, :, i_3, \cdots, i_d) = \mathcal{A}_L(:, :, i_3, \cdots, i_d) \cdot \mathcal{B}_L(:, :, i_3, \cdots, i_d)$;
end
Compute the result of inverse linear transform on $\mathcal{C}_L$
$\mathcal{C} \leftarrow L^{-1}(\mathcal{C}_L)$.

Definition 1.1. ***(Order-d t-product [4, 6])*** *Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times l \times n_3 \times \cdots \times n_d}$, then the tensor-tensor product (t-product) $\mathcal{C} = \mathcal{A} * \mathcal{B}$ is an $n_1 \times l \times n_3 \times \cdots \times n_d$ tensor defined as follows:*

$$\mathcal{C} = \mathcal{A} * \mathcal{B} = \mathrm{bfold}(\mathrm{bcirc}(\mathcal{A}) \cdot \mathrm{bunfold}(\mathcal{B})). \tag{1}$$

The order-d t-product in (1) can be converted to the matrix-matrix multiplication in the transform domain. That is to say,

$$\mathcal{C} = \mathcal{A} *_L \mathcal{B} = L^{-1}(\mathcal{A}_L \bigtriangleup \mathcal{B}_L). \tag{2}$$

The computational procedure of generic order-d t-product is shown in Algorithm 1.

Definition 1.2. ***(Order-d conjugate transpose [6])*** *The conjugate transpose of a tensor $\mathcal{A} \in \mathbb{C}^{n_1 \times n_2 \times n_3 \times \cdots \times n_d}$ is the tensor $\mathcal{A}^* \in \mathbb{C}^{n_2 \times n_1 \times n_3 \times \cdots \times n_d}$ if ${\mathcal{A}^*}_L(:, :, i_3, \cdots, i_d) = \left(\mathcal{A}_L(:, :, i_3, \cdots, i_d)\right)^*$, for all $i_j \in [n_j], j \in \{3, \cdots, d\}$.*

Definition 1.3. ***(order-d identity tensor [6])*** *The order-d identity tensor $\mathcal{I} \in \mathbb{R}^{n \times n \times n_3 \times \cdots \times n_d}$ is the tensor such that $\mathcal{I}_L(:, :, i_3, \cdots, i_d) = \mathrm{I}_n$ for all $i_j \in [n_j], j \in \{3, \cdots, d\}$.*

Definition 1.4. ***(Order-d orthogonal tensor [6])*** *An order-d tensor $\mathcal{Q} \in \mathbb{C}^{n \times n \times n_3 \times \cdots \times n_d}$ is orthogonal if it satisfies $\mathcal{Q}^* *_L \mathcal{Q} = \mathcal{Q} *_L \mathcal{Q}^* = \mathcal{I}$.*

Definition 1.5. ***(Order-d f-diagonal tensor [6])*** *An order-d tensor $\mathcal{A} \in \mathbb{R}^{n \times n \times n_3 \times \cdots \times n_d}$ is called f-diagonal if $\mathcal{A}(:, :, i_3, \cdots, i_d)$ is a diagonal matrix for all $i_j \in [n_j], j \in \{3, \cdots, d\}$.*

Definition 1.6. ***(Order-d f-upper triangular tensor)*** *An order-d tensor $\mathcal{A} \in \mathbb{R}^{n \times n \times n_3 \times \cdots \times n_d}$ is called f-upper triangular if $\mathcal{A}(:, :, i_3, \cdots, i_d)$ is a an upper triangular matrix for all $i_j \in [n_j], j \in \{3, \cdots, d\}$.*

Definition 1.7. ***(Order-d Gaussian random tensor)*** *$\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is called a Gaussian random tensor, if $\mathcal{A}_L(:, :, i_3, \cdots, i_d)$ satisfy the standard normal distribution for all $i_j \in [n_j], j \in \{3, \cdots, d\}$.*

Definition 1.8. ***(Order-d t-QR decomposition [4])*** *Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times \cdots \times n_d}$, then it can be decomposed as*

$$\mathcal{A} = \mathcal{Q} *_L \mathcal{R}, \tag{3}$$

where $\mathcal{Q} \in \mathbb{R}^{n_1 \times n_1 \times n_3 \times \cdots \times n_d}$ is an orthogonal tensor, $\mathcal{R} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times \cdots \times n_d}$ is f-upper triangular.

The computational procedure of generic order-d t-QR decomposition is presented in Algorithm 2.

Definition 1.9. ***(Order-d t-SVD [6])*** *Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times \cdots \times n_d}$, then it can be factorized as*

$$\mathcal{A} = \mathcal{U} *_L \mathcal{S} *_L \mathcal{V}^*, \tag{4}$$

where $\mathcal{U} \in \mathbb{R}^{n_1 \times n_1 \times n_3 \times \cdots \times n_d}$, $\mathcal{V} \in \mathbb{R}^{n_2 \times n_2 \times n_3 \times \cdots \times n_d}$ are orthogonal, $\mathcal{S} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times \cdots \times n_d}$ is a f-diagonal tensor which has the property that $\mathcal{S}_{i_1 \cdots i_d} = 0$ unless $n_1 = n_2$.

The computational procedure of generic order-d t-SVD is presented in Algorithm 3.

Algorithm 2: Generic order-d t-QR decomposition, tqr-L($\boldsymbol{\mathcal{A}}, L$).

Input: $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and corresponding matrices $\{\mathrm{U}_{n_i}\}_{i=3}^{d}$ of linear transforms L.
Output: t-QR components $\boldsymbol{\mathcal{Q}}$ and $\boldsymbol{\mathcal{R}}$ of $\boldsymbol{\mathcal{A}}$.
Compute the result of linear transform on $\boldsymbol{\mathcal{A}}$
$\boldsymbol{\mathcal{A}}_L \leftarrow L(\boldsymbol{\mathcal{A}})$;
Compute the matrix slice of $\boldsymbol{\mathcal{Q}}_L$ and $\boldsymbol{\mathcal{R}}_L$ from $\boldsymbol{\mathcal{A}}_L$ by
for $i_3 \in \{1, \cdots, n_3\}, \cdots, i_d \in \{1, \cdots, n_d\}$ **do**
    $[\mathrm{Q}, \mathrm{R}] = \mathrm{qr}\big(\boldsymbol{\mathcal{A}}_L(:, :, i_3, \cdots, i_d), 0\big)$,
    $\boldsymbol{\mathcal{Q}}_L(:, :, i_3, \cdots, i_d) = \mathrm{Q}$, $\boldsymbol{\mathcal{R}}_L(:, :, i_3, \cdots, i_d) = \mathrm{R}$;
end
Compute the result of inverse linear transform on $\boldsymbol{\mathcal{Q}}_L$ and $\boldsymbol{\mathcal{R}}_L$
$\boldsymbol{\mathcal{Q}} \leftarrow L^{-1}(\boldsymbol{\mathcal{Q}}_L)$ and $\boldsymbol{\mathcal{R}} \leftarrow L^{-1}(\boldsymbol{\mathcal{R}}_L)$.

Algorithm 3: Generic order-d t-SVD [6], tsvd-L($\boldsymbol{\mathcal{A}}, L$).

Input: $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and corresponding matrices $\{\mathrm{U}_{n_i}\}_{i=3}^{d}$ of linear transforms L.
Output: t-SVD components $\boldsymbol{\mathcal{U}}$, $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{V}}$ of $\boldsymbol{\mathcal{A}}$.
Compute the result of linear transform on $\boldsymbol{\mathcal{A}}$
$\boldsymbol{\mathcal{A}}_L \leftarrow L(\boldsymbol{\mathcal{A}})$;
Compute the matrix slice of $\boldsymbol{\mathcal{U}}_L$, $\boldsymbol{\mathcal{S}}_L$ and $\boldsymbol{\mathcal{V}}_L$ from $\boldsymbol{\mathcal{A}}_L$ by
for $i_3 \in \{1, \cdots, n_3\}, \cdots, i_d \in \{1, \cdots, n_d\}$ **do**
    $[\mathrm{U}, \mathrm{S}, \mathrm{V}] = \mathrm{svd}\big(\boldsymbol{\mathcal{A}}_L(:, :, i_3, \cdots, i_d)\big)$,
    $\boldsymbol{\mathcal{U}}_L(:, :, i_3, \cdots, i_d) = \mathrm{U}$, $\boldsymbol{\mathcal{S}}_L(:, :, i_3, \cdots, i_d) = \mathrm{S}$, $\boldsymbol{\mathcal{V}}_L(:, :, i_3, \cdots, i_d) = \mathrm{V}$;
end
Compute the result of inverse linear transform on $\boldsymbol{\mathcal{U}}_L$, $\boldsymbol{\mathcal{S}}_L$ and $\boldsymbol{\mathcal{V}}_L$
$\boldsymbol{\mathcal{U}} \leftarrow L^{-1}(\boldsymbol{\mathcal{U}}_L)$, $\boldsymbol{\mathcal{S}} \leftarrow L^{-1}(\boldsymbol{\mathcal{S}}_L)$ and $\boldsymbol{\mathcal{V}} \leftarrow L^{-1}(\boldsymbol{\mathcal{V}}_L)$.

Definition 1.10. ***(Order-d t-SVD rank [6])*** *For any $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, let $\boldsymbol{\mathcal{S}}$ be from the t-SVD component of $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{U}} *_L \boldsymbol{\mathcal{S}} *_L \boldsymbol{\mathcal{V}}^*$. Then the t-SVD rank of $\boldsymbol{\mathcal{A}}$ is defined as*

$$\begin{aligned}\mathrm{rank}_{tsvd}(\boldsymbol{\mathcal{A}}) &= \sharp\big\{i : \boldsymbol{\mathcal{S}}(i, i, :, \cdots, :) \neq \mathbf{0}\big\}, \\ &= \max_{i_3 \in [n_3], \cdots, i_d \in [n_d]} \mathrm{rank}\left(\boldsymbol{\mathcal{A}}_L(:, :, i_3, \cdots, i_d)\right),\end{aligned}$$

where $\sharp$ denotes the cardinality of a set.

Remark 1.1. *Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ with $\mathrm{rank}_{tsvd}(\boldsymbol{\mathcal{X}}) = r$, the skinny t-SVD of tensor $\boldsymbol{\mathcal{X}}$ is $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{U}} *_L \boldsymbol{\mathcal{S}} *_L \boldsymbol{\mathcal{V}}^*$, where $\boldsymbol{\mathcal{U}} \in \mathbb{R}^{n_1 \times r \times n_3 \times \cdots \times n_d}$ and $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{n_2 \times r \times n_3 \times \cdots \times n_d}$ satisfy $\boldsymbol{\mathcal{U}}^* *_L \boldsymbol{\mathcal{U}} = \boldsymbol{\mathcal{I}}$ and $\boldsymbol{\mathcal{V}}^* *_L \boldsymbol{\mathcal{V}} = \boldsymbol{\mathcal{I}}$, and $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{r \times r \times n_3 \times \cdots \times n_d}$ is a f-diagonal tensor which has the property that $\boldsymbol{\mathcal{S}}_{i_1 \cdots i_d} = 0$ unless $n_1 = n_2$.*

Definition 1.11. ***(Order-d tensor spectral norm [6])*** *The spectral norm of $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is defined as $\|\boldsymbol{\mathcal{A}}\| := \|\mathrm{bdiag}(\boldsymbol{\mathcal{A}}_L)\|$.*

Definition 1.12. ***(Order-d TNN [6])*** *Let $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and $m = \min(n_1, n_2)$, then the tensor nuclear norm (TNN) of $\boldsymbol{\mathcal{A}}$ is defined as*

$$\|\boldsymbol{\mathcal{A}}\|_{\star, L} := \frac{1}{\rho} \|\,\mathrm{bdiag}(\boldsymbol{\mathcal{A}}_L)\|_\star \tag{5}$$

$$= \frac{1}{\rho} \sum_{i=1}^{m} \sum_{i_3=1}^{n_3} \cdots \sum_{i_d=1}^{n_d} \boldsymbol{\mathcal{S}}_L(i, i, i_3, \cdots, i_d), \tag{6}$$

where $\rho > 0$ is a positive constant determined by the invertible linear transforms L, and the entries on the diagonal of $\boldsymbol{\mathcal{S}}_L(:, :, i_3, \cdots, i_d)$ denote the singular values of $\boldsymbol{\mathcal{A}}_L(:, :, i_3, \cdots, i_d)$.

Definition 1.13. *(Order-d WTNN [6]) Let $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and $m = \min(n_1, n_2)$, then the weighted tensor nuclear norm (WTNN) of $\mathcal{A}$ is defined as*

$$\|\mathcal{A}\|_{\mathcal{W},L} := \frac{1}{\rho}\| \operatorname{bdiag}(\mathcal{A}_L)\|_w \tag{7}$$

$$= \frac{1}{\rho}\sum_{i_3=1}^{n_3}\cdots\sum_{i_d=1}^{n_d}\left\|\mathcal{A}_L(:,:,i_3,\cdots,i_d)\right\|_w \tag{8}$$

$$= \frac{1}{\rho}\sum_{i=1}^{m}\sum_{i_3=1}^{n_3}\cdots\sum_{i_d=1}^{n_d}\mathcal{W}(i,i,i_3,\cdots,i_d)\ \mathcal{S}_L(i,i,i_3,\cdots,i_d), \tag{9}$$

where $\rho > 0$ is a positive constant determined by the linear transforms L, $\mathcal{W}(i,i,i_3,\cdots,i_p) \geq 0$ is the weight parameter, $w = \operatorname{diag}(\operatorname{bdiag}(\mathcal{W}))$, and the entries on the diagonal of $\mathcal{S}_L(:,:,i_3,\cdots,i_d)$ denote the singular values of $\mathcal{A}_L(:,:,i_3,\cdots,i_d)$.

Definition 1.14. *(Order-d WTSN) Let $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and $m = \min(n_1, n_2)$, then the weighted tensor Schatten-p norm (WTSN) of $\mathcal{A}$ is defined as*

$$\|\mathcal{A}\|_{\mathcal{W},\mathcal{S}_p} := \left(\frac{1}{\rho}\left\| \operatorname{bdiag}(\mathcal{A}_L)\right\|_{w,\mathrm{S}_p}^p\right)^{1/p} \tag{10}$$

$$= \left(\frac{1}{\rho}\sum_{i_3=1}^{n_3}\cdots\sum_{i_d=1}^{n_d}\left\|\mathcal{A}_L(:,:,i_3,\cdots,i_d)\right\|_{w,\mathrm{S}_p}^p\right)^{\frac{1}{p}} \tag{11}$$

$$= \left(\frac{1}{\rho}\sum_{i=1}^{m}\sum_{i_3=1}^{n_3}\cdots\sum_{i_d=1}^{n_d}\mathcal{W}(i,i,i_3,\cdots,i_p)\left|\mathcal{S}_L(i,i,i_3,\cdots,i_p)\right|^p\right)^{\frac{1}{p}}, \tag{12}$$

where $\rho > 0$ is a positive constant determined by the linear transforms L, $\mathcal{W}(i,i,i_3,\cdots,i_p) \geq 0$ is the weight parameter, $w = \operatorname{diag}(\operatorname{bdiag}(\mathcal{W}))$, and the entries on the diagonal of $\mathcal{S}_L(:,:,i_3,\cdots,i_d)$ denote the singular values of $\mathcal{A}_L(:,:,i_3,\cdots,i_d)$.

The WTSN defined in (10) can be equivalently reformulated as follows:

$$\|\mathcal{A}\|_{\mathcal{W},\mathcal{S}_p}^p = \frac{1}{\rho}\sum_{i_3=1}^{n_3}\cdots\sum_{i_d=1}^{n_d}\operatorname{tr}\left(\mathrm{W}^{(i_3,\cdots,i_p)}\left|\mathrm{S}_L{}^{(i_3,\cdots,i_p)}\right|^p\right). \tag{13}$$

Remark 1.2. *Throughout the article, the constant ρ appeared in the key definition and theorem is the constant ρ obtained when corresponding matrices of the invertible linear transform L satisfy the equation:*

$$\begin{aligned}&(\mathrm{U}_{n_p}\otimes\mathrm{U}_{n_{p-1}}\otimes\cdots\otimes\mathrm{U}_{n_3})(\mathrm{U}_{n_p}^*\otimes\mathrm{U}_{n_{p-1}}^*\otimes\cdots\otimes\mathrm{U}_{n_3}^*)\\&=(\mathrm{U}_{n_p}^*\otimes\mathrm{U}_{n_{p-1}}^*\otimes\cdots\otimes\mathrm{U}_{n_3}^*)(\mathrm{U}_{n_p}\otimes\mathrm{U}_{n_{p-1}}\otimes\cdots\otimes\mathrm{U}_{n_3})\\&=\rho\mathrm{I}_{n_3n_4\cdots n_p}.\end{aligned} \tag{14}$$

The constant ρ is related to the invertible linear transform L. For instance, if the invertible transform matrices $\{\mathrm{U}_{n_i}\}_{i=3}^d$ satisfy: $\mathrm{U}_{n_3}\times\mathrm{U}_{n_3}^* = \mathrm{U}_{n_3}^*\times\mathrm{U}_{n_3} = n_3\mathrm{I}_{n_3}, \cdots, \mathrm{U}_{n_d}\times\mathrm{U}_{n_d}^* = \mathrm{U}_{n_d}^*\times\mathrm{U}_{n_d} = n_d\mathrm{I}_{n_d}$, then ρ equals to $n_3\times\cdots\times n_d$.

2. ORDER-D RT-SVD SCHEME

In this section, we propose the generic randomized tensor Singular Value Decomposition (rt-SVD) method, which can be viewed as a flexible extension of the matrix randomized SVD (r-SVD) [7]. The generic order-d rt-SVD approach mainly incorporates the randomized technique into the generic order-d t-SVD as a result of better efficiency.

The simplified version of generic order-d rt-SVD is presented in Algorithm 4. Following a similar acceleration technique to matrix r-SVD, the core of generic order-d rt-SVD method is to find a good approximate factorization of tensor $\mathcal{A} \in \mathbb{R}^{n_1\times\cdots\times n_d}$, $\mathcal{U}*_L\mathcal{S}*_L\mathcal{V}^*$. This approximation can be implemented by multiplying $\mathcal{A}$ with a random tensor $\mathcal{G}$ on its right side, and then obtaining an orthogonal subspace basis tensor $\mathcal{Q}$ such that $\mathcal{A} \approx \mathcal{Q}*_L\mathcal{Q}^**_L\mathcal{A}$. Specifically, one can capture the main actions for the column space of $\mathcal{A} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ via $\mathcal{Y} = \mathcal{A}*_L\mathcal{G}$ with $\mathcal{G}$ an $n_2 \times (k+q) \times n_3 \times \cdots \times n_d$ Gaussian random tensor.

Algorithm 4: Generic order-d rt-SVD (simplified version).

Input: $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1\times\cdots\times n_d}$, truncation term $k < \min(n_1, n_2)$, oversampling parameter: $q > 0$, $l = k + q$ and corresponding matrices $\{\mathrm{U}_{n_i}\}_{i=3}^{d}$ of linear transforms L.

Output: $\boldsymbol{\mathcal{U}} \in \mathbb{R}^{n_1\times k\times n_3\times\cdots\times n_d}$,$\boldsymbol{\mathcal{S}} \in \mathbb{R}^{k\times k\times n_3\times\cdots\times n_d}$, $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{n_2\times k\times n_3\times\cdots\times n_d}$.

Generate a Gaussian random tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{n_2\times l\times n_3\times\cdots\times n_d}$;
Construct a random projection of tensor $\boldsymbol{\mathcal{A}}$ as $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{A}} *_L \boldsymbol{\mathcal{G}}$;
Form the tensor $\boldsymbol{\mathcal{Q}} \in \mathbb{R}^{n_1\times l\times n_3\times\cdots\times n_d}$ by using t-QR decomposition of $\boldsymbol{\mathcal{Y}}$;
Construct a tensor $\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{Q}}^* *_L \boldsymbol{\mathcal{A}}$, whose size is $l \times n_2 \times n_3 \times \cdots \times n_d$;
Compute t-SVD of $\boldsymbol{\mathcal{B}}$, truncate it with target truncation term k, and obtain $\boldsymbol{\mathcal{U}}_k, \boldsymbol{\mathcal{S}}_k$, and $\boldsymbol{\mathcal{V}}_k$;
Form the rt-SVD components of $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{U}} = (\boldsymbol{\mathcal{Q}} *_L \boldsymbol{\mathcal{U}}_k), \boldsymbol{\mathcal{S}} = \boldsymbol{\mathcal{S}}_k, \boldsymbol{\mathcal{V}} = \boldsymbol{\mathcal{V}}_k$.

Algorithm 5: PowerMethod $(\mathrm{A}, \mathrm{W}, \eta)$ [7,8].

Input: $\mathrm{A} \in \mathbb{R}^{m\times n}$, $\mathrm{W} \in \mathbb{R}^{n\times k}$, and the number of power iterations η.

```
Initialize: Y_1 = AW;
for j = 1, 2, ..., η do
    Q_j = qr(Y_j, 0); // qr(·) is QR factorization
    Y_{j+1} = A(A^⊤ Q_j);
end
return: PowerMethod(A, W, η) = Q_j.
```

Algorithm 6: Randomized SVD method using power iteration, r-svd$(\mathrm{A}, \mathrm{G}, k, q, p)$.

Input: $\mathrm{A} \in \mathbb{R}^{n_1\times n_2}$, truncation term $k < \min(n_1, n_2)$, oversampling parameter: $q > 0$, $p \geq 0$, $l = k + q$ and Gaussian random matrix $\mathrm{G} \in \mathbb{R}^{n_2\times l}$.

Output: r-SVD components $\mathrm{U} \in \mathbb{R}^{n_1\times k}$, $\mathrm{S} \in \mathbb{R}^{k\times k}$, $\mathrm{V} \in \mathbb{R}^{n_2\times k}$ of A.

$\mathrm{Q}_1 = \mathrm{PowerMethod}\left(\mathrm{A}, \mathrm{G}, p\right)$; // PowerMethod(·) see Algorithm 5
Obtain the small matrix R using the QR factorization $\left[\mathrm{Q}_2, \mathrm{R}\right] = \mathrm{qr}\left(\mathrm{A}^* \cdot \mathrm{Q}_1, 0\right)$;
Take the SVD of matrix R, i.e., $\left[\mathrm{U}_1, \Lambda_1, \mathrm{V}_1\right] = \mathrm{svd}(\mathrm{R})$;
Approximate the SVD components of A using the results of the SVD of R
$\mathrm{U}_k = \mathrm{Q}_1\mathrm{V}_1, \Lambda_k = \Lambda_1, \mathrm{V}_k = \mathrm{Q}_2\mathrm{U}_1$;
Extract components corresponding to the k largest singular values
$\mathrm{U} = \mathrm{U}_k(:, 1:k),\ \mathrm{S} = \Lambda_k(1:k, 1:k),\ \mathrm{V} = \mathrm{V}_k(:, 1:k)$.

Algorithm 7: Generic order-d rt-SVD using power iteration, rtsvd-L$(\boldsymbol{\mathcal{A}}, L, k, q, p)$.

Input: $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1\times\cdots\times n_d}$, truncation term $k < \min(n_1, n_2)$, oversampling parameter: $q > 0$, $p \geq 0$ and corresponding matrices $\{\mathrm{U}_{n_i}\}_{i=3}^{d}$ of linear transforms L.

Output: $\boldsymbol{\mathcal{U}} \in \mathbb{R}^{n_1\times k\times n_3\times\cdots\times n_d}$,$\boldsymbol{\mathcal{S}} \in \mathbb{R}^{k\times k\times n_3\times\cdots\times n_d}$, $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{n_2\times k\times n_3\times\cdots\times n_d}$.

Set $l = k + q$ and initialize a Gaussian random tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{n_2\times l\times n_3\times\cdots\times n_d}$;
Compute the result of linear transform on $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{G}}$
$\boldsymbol{\mathcal{A}}_L \leftarrow L(\boldsymbol{\mathcal{A}}), \boldsymbol{\mathcal{G}}_L \leftarrow L(\boldsymbol{\mathcal{G}})$;
Compute the matrix slice of $\boldsymbol{\mathcal{U}}_L$, $\boldsymbol{\mathcal{S}}_L$ and $\boldsymbol{\mathcal{V}}_L$ from $\boldsymbol{\mathcal{A}}_L$ by
for $i_3 \in \{1, \cdots, n_3\}, \cdots, i_d \in \{1, \cdots, n_d\}$ **do**
  $[\mathrm{U}, \mathrm{S}, \mathrm{V}] = \text{r-svd}\left(\boldsymbol{\mathcal{A}}_L(:, :, i_3, \cdots, i_d), \boldsymbol{\mathcal{G}}_L(:, :, i_3, \cdots, i_d), k, q, p\right)$, // r-svd(·) see Algorithm 6
  $\boldsymbol{\mathcal{U}}_L(:, :, i_3, \cdots, i_d) = \mathrm{U}, \boldsymbol{\mathcal{S}}_L(:, :, i_3, \cdots, i_d) = \mathrm{S}, \boldsymbol{\mathcal{V}}_L(:, :, i_3, \cdots, i_d) = \mathrm{V}$.
end
Compute the result of inverse linear transform on $\boldsymbol{\mathcal{U}}_L$, $\boldsymbol{\mathcal{S}}_L$ and $\boldsymbol{\mathcal{V}}_L$
$\boldsymbol{\mathcal{U}} \leftarrow L^{-1}(\boldsymbol{\mathcal{U}}_L), \boldsymbol{\mathcal{S}} \leftarrow L^{-1}(\boldsymbol{\mathcal{S}}_L)$ and $\boldsymbol{\mathcal{V}} \leftarrow L^{-1}(\boldsymbol{\mathcal{V}}_L)$.

Algorithm 8: Generalized Soft-Thresholding (GST) [9, 10].

Input: s, w, p, J.
Output: $\mathrm{GST}\big(s, w, p, J\big)$.

$\delta_p^{GST}(w) = [2w(1-p)]^{\frac{1}{2-p}} + wp[2w(1-p)]^{\frac{p-1}{2-p}}$;
if $|s| \leq \delta_p^{GST}(w)$ **then**
  $\mathrm{GST}\big(s, w, p, J\big) = 0$;
else
  $j = 0, x^{(j)} = |s|$;
  for $j = 0, 1, \cdots, J$ **do**
    $x^{(j+1)} = |s| - wp(x^{(j)})^{p-1}$;
    $j = j + 1$;
  end
  $\mathrm{GST}\big(s, w, p, J\big) = \mathrm{sgn}(s)x^{(j)}$;
end

As a result, $\boldsymbol{\mathcal{Y}}$ is the tensor of size $n_1 \times (k+q) \times n_3 \times \cdots \times n_d$, in which k is the desired t-SVD rank and q denotes a small oversampling parameter. Eventually, $\boldsymbol{\mathcal{Q}}$ can be achieved from the tensor QR decomposition (t-QR) of $\boldsymbol{\mathcal{Y}}$.

To further improve the accuracy of randomized approximation of $\boldsymbol{\mathcal{A}}$, we can additionally apply the power iteration scheme [7], which multiplies alternately with $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{A}}^*$, i.e., $(\boldsymbol{\mathcal{A}} *_L \boldsymbol{\mathcal{A}}^*)^p *_L \boldsymbol{\mathcal{A}}$, where p is a nonnegative integer. In other word, we replace the Step 2 of Algorithm 4 with

$$\boldsymbol{\mathcal{Y}} = (\boldsymbol{\mathcal{A}} *_L \boldsymbol{\mathcal{A}}^*)^p *_L \boldsymbol{\mathcal{A}} *_L \boldsymbol{\mathcal{G}}.$$

The power iteration scheme works efficiently when the singular values of $\boldsymbol{\mathcal{A}}_L(:, :, i_3, \cdots, i_d)$ decay at a comparable rate. The generic order-d rt-SVD using power iteration scheme is presented in Algorithm 7, which is built on Algorithm 5 and 6.

3. THE PROXIMAL OPERATOR OF ORDER-D WTSN

In this section, in virtue of generalized soft-thresholding (GST) algorithm [9, 10], we provide the calculation method of the proximal operator of order-d WTSN

$$\arg\min_{\boldsymbol{\mathcal{X}}} \tau \|\boldsymbol{\mathcal{X}}\|_{\boldsymbol{\mathcal{W}}, \boldsymbol{\mathcal{S}}_p}^p + \frac{1}{2}\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Z}}\|_F^2. \tag{15}$$

Definition 3.1. ***(Order-d WTSN proximal operator)*** *Let $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{U}} *_L \boldsymbol{\mathcal{S}} *_L \boldsymbol{\mathcal{V}}^*$ be the t-SVD of $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. For any $\tau > 0$, the order-d WTSN operator is defined as follows*

$$\boldsymbol{\mathcal{D}}_{\boldsymbol{\mathcal{W}}, p, \tau}(\boldsymbol{\mathcal{A}}) = \boldsymbol{\mathcal{U}} *_L \boldsymbol{\mathcal{S}}_{\boldsymbol{\mathcal{W}}, p, \tau} *_L \boldsymbol{\mathcal{V}}^*, \tag{16}$$

where

$$\boldsymbol{\mathcal{S}}_{\boldsymbol{\mathcal{W}}, p, \tau} = L^{-1}\big(GST(\boldsymbol{\mathcal{S}}_L, \tau\boldsymbol{\mathcal{W}}, p, J)\big),$$

in which J is the number of iterations of GST algorithm, p $(0 < p < 1)$ represents the adjustable parameters appeared in order-d WTSN, and $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ denotes the weight parameter composed of an order-d f-diagonal tensor.

The computational procedure of generic order-d WTSN proximal operator is presented in Algorithm 9.

Lemma 3.1. *[10] Let the SVD of $\mathrm{Y} \in \mathbb{R}^{n_1 \times n_2}$ be $\mathrm{Y} = \mathrm{U}\Sigma\mathrm{V}^\top$ with $\Sigma = \mathrm{diag}\{\sigma_1, \ldots, \sigma_m\}$. For any $\tau \geq 0$ and $0 \leq w_1 \leq w_2 \leq \cdots \leq w_m$ ($m = \min\{n_1, n_2\}$), a global optimal solution to the optimization problem*

$$\min_{\mathrm{X}} \tau \|\mathrm{X}\|_{w, S_p}^p + \frac{1}{2}\|\mathrm{X} - \mathrm{Y}\|_F^2$$

is given by

$$\mathcal{D}_{\mathbf{w}, \tau, p}(\mathrm{Y}) = \mathrm{U} \cdot \mathrm{S}_{\mathbf{w}, \tau, p}(\mathrm{Y}) \cdot \mathrm{V}^{\mathrm{T}},$$

where $\mathrm{S}_{\mathbf{w}, \tau, p}(\mathrm{Y}) = \mathrm{diag}\Big\{GST\big(\sigma_i(\mathrm{Y}), \tau w_i, p, J\big), i = 1, \cdots, m\Big\}$.

Algorithm 9: Generic order-d WTSN proximal operator.

Input: $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, weight parameter: $\mathcal{W} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, truncation term: $k < \min(n_1, n_2)$, the number of GST and PowerMethod iteration: $\alpha, \beta, \tau > 0$, $0 < p < 1$, oversampling parameter: $q > 0$, and corresponding matrices $\{\mathrm{U}_{n_i}\}_{i=3}^{d}$ of invertible linear transforms L.

Output: $\mathcal{D}_{\mathcal{W},p,\tau}(\mathcal{A}) = \mathcal{U} *_L \mathcal{S}_{\mathcal{W},p,\tau} *_L \mathcal{V}^*$.

Compute the rt-SVD or t-SVD components $\mathcal{U}$,$\mathcal{S}$ and $\mathcal{V}$ of $\mathcal{A}$

if *utilize the rt-SVD scheme* **then**

  $[\mathcal{U}, \mathcal{S}, \mathcal{V}] = \text{rtsvd-L}(\mathcal{A}, L, k, q, \beta)$; // rtsvd-L($\cdot$) see Algorithm 7

end

else if *utilize the t-SVD scheme* **then**

  $[\mathcal{U}, \mathcal{S}, \mathcal{V}] = \text{tsvd-L}(\mathcal{A}, L)$; // tsvd-L($\cdot$) see Algorithm 3

end

Compute the matrix slice of $\mathcal{C}_L$ by

for $i_3 \in \{1, \cdots, n_3\}, \cdots, i_d \in \{1, \cdots, n_d\}$ **do**

  $diagS = \text{GST}\Big(\text{diag}\big(\mathcal{S}_L(:,:,i_3,\cdots,i_d)\big), \tau \text{diag}\big(\mathcal{W}(:,:,i_3,\cdots,i_d)\big), p, \alpha\Big)$; // GST($\cdot$) see Algorithm 8

  $\mathcal{C}_L(:,:,i_3,\cdots,i_d) = \text{diag}(diagS)$;

end

Compute the result of inverse linear transform on $\mathcal{C}_L$

$\mathcal{S}_{\mathcal{W},p,\tau} \leftarrow L^{-1}(\mathcal{C}_L)$.

Compute $\mathcal{D}_{\mathcal{W},p,\tau}(\mathcal{A}) = \text{tpro-L}\big(\text{tpro-L}(\mathcal{U}, \mathcal{S}_{\mathcal{W},p,\tau}), \mathcal{V}^*\big)$. // tpro-L($\cdot$) see Algorithm 1

Theorem 3.1. *Let $m = \min(n_1, n_2)$. For any $\tau > 0$ and $\mathcal{Z} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, then the order-$d$ WTSN operator (16) obeys*

$$\mathcal{D}_{\mathcal{W},p,\tau}(\mathcal{Z}) = \arg\min_{\mathcal{X}} \tau \|\mathcal{X}\|_{\mathcal{W},\mathcal{S}_p}^p + \frac{1}{2}\|\mathcal{X} - \mathcal{Z}\|_F^2, \tag{17}$$

if the weight parameter satisfies $0 \leq \mathcal{W}(1,1,i_3,\cdots,i_d) \leq \cdots \leq \mathcal{W}(m,m,i_3,\cdots,i_d)$, for all $i_j \in [n_j], j \in \{3,\cdots,d\}$.

Proof. Based on the definition of order-d WTSN, on the one hand, we have

$$\tau\|\mathcal{X}\|_{\mathcal{W},\mathcal{S}_p}^p = \frac{1}{\rho}\sum_{i_3=1}^{n_3}\cdots\sum_{i_d=1}^{n_d} \tau \big\|\mathcal{X}_L(:,:,i_3,\cdots,i_d)\big\|_{w,S_p}^p, \tag{18}$$

where $w = \text{diag}\big(\mathcal{W}(:,:,i_3,\cdots,i_d)\big)$. Utilizing the property (14), on the other hand, we have the following important equations:

$$\big\|\mathcal{A}\big\|_F = \frac{1}{\sqrt{\rho}}\big\|\,\text{bdiag}(\mathcal{A}_L)\big\|_F,$$
$$\big\langle \mathcal{A}, \mathcal{B}\big\rangle = \frac{1}{\rho}\big\langle\, \text{bdiag}(\mathcal{A}_L), \text{bdiag}(\mathcal{B}_L)\big\rangle,$$

thus

$$\begin{aligned}\frac{1}{2}\big\|\mathcal{X} - \mathcal{Z}\big\|_F^2 &= \frac{1}{2\rho}\big\|\,\text{bdiag}(\mathcal{X}_L) - \text{bdiag}(\mathcal{Z}_L)\big\|_F^2 \\ &= \frac{1}{2\rho}\sum_{i_3=1}^{n_3}\cdots\sum_{i_d=1}^{n_d}\big\|\mathcal{X}_L(:,:,i_3,\cdots,i_d) - \mathcal{Z}_L(:,:,i_3,\cdots,i_d)\big\|_F^2.\end{aligned} \tag{19}$$

Then, the problem (17) is equivalent to

$$\begin{aligned}\arg\min_{\mathcal{X}} \frac{1}{\rho}\sum_{i_3=1}^{n_3}\cdots\sum_{i_d=1}^{n_d}\Big(&\tau\big\|\mathcal{X}_L(:,:,i_3,\cdots,i_d)\big\|_{w,S_p}^p \\ &+ \frac{1}{2}\big\|\mathcal{X}_L(:,:,i_3,\cdots,i_d) - \mathcal{Z}_L(:,:,i_3,\cdots,i_d)\big\|_F^2\Big).\end{aligned} \tag{20}$$

Since $0 \leq \mathcal{W}(1,1,i_3,\cdots,i_d) \leq \cdots \leq \mathcal{W}(m,m,i_3,\cdots,i_d)$, the $(i_3,\cdots,i_d)$-th matrix slice of $L\big(\mathcal{D}_{\mathcal{W},p,\tau}(\mathcal{Z})\big)$ solves the $(i_3,\cdots,i_d)$-th subproblem of (20) in virtue of Lemma 3.1. Hence, $\mathcal{D}_{\mathcal{W},p,\tau}(\mathcal{Z})$ solves the problem (17). $\square$

Algorithm 10: FFT based order-d t-product, tpro-fft($\mathcal{A}, \mathcal{B}$).

Input: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times \cdots \times n_d}$, $\mathcal{B} \in \mathbb{R}^{n_2 \times l \times n_3 \times \cdots \times n_d}$.

Output: $\mathcal{C} = \mathcal{A} * \mathcal{B} \in \mathbb{R}^{n_1 \times l \times n_3 \times \cdots \times n_d}$.

Compute the result of FFT on $\mathcal{A}$ and $\mathcal{B}$

for $i = 3, 4, \cdots, d$ **do**

  $\mathcal{A}_\mathrm{f} \leftarrow \mathrm{fft}(\mathcal{A}, [\,], i)$, $\mathcal{B}_\mathrm{f} \leftarrow \mathrm{fft}(\mathcal{B}, [\,], i)$;

end

Compute the matrix slice of $\mathcal{C}_\mathrm{f}$ for given index by

for $i_3 \in \left\{2, \cdots, \lceil \frac{n_3+1}{2} \rceil\right\}, \cdots, i_d \in \left\{2, \cdots, \lceil \frac{n_d+1}{2} \rceil\right\}$,

$i_3^{'} \in \left\{1, \cdots, \lceil \frac{n_3+1}{2} \rceil\right\}, \cdots, i_{d-1}^{'} \in \left\{1, \cdots, \lceil \frac{n_{d-1}+1}{2} \rceil\right\}$ **do**

$$\begin{cases} \mathrm{C_f}^{(1,1,1,\cdots,1,1)} = \mathrm{A_f}^{(1,1,1,\cdots,1,1)} \cdot \mathrm{B_f}^{(1,1,1,\cdots,1,1)}, \\ \mathrm{C_f}^{(i_3,1,1,\cdots,1,1)} = \mathrm{A_f}^{(i_3,1,1,\cdots,1,1)} \cdot \mathrm{B_f}^{(i_3,1,1,\cdots,1,1)}, \\ \mathrm{C_f}^{(i_3^{'},i_4,1,\cdots,1,1)} = \mathrm{A_f}^{(i_3^{'},i_4,1,\cdots,1,1)} \cdot \mathrm{B_f}^{(i_3^{'},i_4,1,\cdots,1,1)}, \\ \mathrm{C_f}^{(i_3^{'},i_4^{'},i_5,\cdots,1,1)} = \mathrm{A_f}^{(i_3^{'},i_4^{'},i_5,\cdots,1,1)} \cdot \mathrm{B_f}^{(i_3^{'},i_4^{'},i_5,\cdots,1,1)}, \\ \cdots\cdots \\ \mathrm{C_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)} = \mathrm{A_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)} \cdot \mathrm{B_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}. \end{cases}$$

end

Compute the remaining matrix slice of $\mathcal{C}_\mathrm{f}$ by conjugate symmetry (22);

Compute the result of inverse FFT on $\mathcal{C}_\mathrm{f}$

for $i = d, d-1, \cdots, 3$ **do**

  $\mathcal{C} \leftarrow \mathrm{ifft}(\mathcal{C}_\mathrm{f}, [\,], i)$.

end

4. FFT BASED RELEVANT ALGORITHMS

In this section, we present a series of commonly-used algorithms that implement in the Fourier domain. These algorithms can resort to the conjugate symmetry of FFT to reduce the computational cost.

Remark 4.1. *When we utilize the Fast Fourier Transform (FFT) as the invertible linear transform* L*, the block circulant matrix of* $\mathcal{A}$ *and the block diagonal matrix of* $\mathcal{A}_\mathrm{f}$ *have the following relationship:*

$$(\tilde{\mathrm{F}} \otimes \mathrm{I}_{n_1}) \cdot \mathrm{bcirc}(\mathcal{A}) \cdot (\tilde{\mathrm{F}}^{-1} \otimes \mathrm{I}_{n_2}) = \mathrm{bdiag}(\mathcal{A}_\mathrm{f}), \tag{21}$$

where $\tilde{\mathrm{F}} = \mathrm{F}_{n_d} \otimes \mathrm{F}_{n_{d-1}} \otimes \cdots \otimes \mathrm{F}_{n_3}$*,* $\tilde{\mathrm{F}}^{-1} = \mathrm{F}_{n_d}^{-1} \otimes \mathrm{F}_{n_{d-1}}^{-1} \otimes \cdots \otimes \mathrm{F}_{n_3}^{-1}$*,* $\mathrm{F}_{n_d} \in \mathbb{C}^{n_d \times n_d}$ *denotes the DFT matrix, and* $(\mathrm{F}_{n_d} \otimes \mathrm{F}_{n_{d-1}} \otimes \cdots \otimes \mathrm{F}_{n_3})/\sqrt{n_d n_{d-1} \cdots n_3}$ *is orthogonal. By using the property of real symmetric circulant matrix (see the Definition 1 in [11]), we have*

$$\begin{cases} \mathrm{A_f}^{(1,1,1,\cdots,1,1)} \in \mathbb{R}^{n_1 \times n_2}, \\ \mathrm{conj}(\mathrm{A_f}^{(i_3,1,\cdots,1)}) = \mathrm{A_f}^{(n_3-i_3+2,1,\cdots,1)}, \\ \mathrm{conj}(\mathrm{A_f}^{(i_3^{'},i_4,1,\cdots,1)}) = \mathrm{A_f}^{(n_3^{'},n_4-i_4+2,1,\cdots,1)}, \\ \mathrm{conj}(\mathrm{A_f}^{(i_3^{'},i_4^{'},i_5,1,\cdots,1)}) = \mathrm{A_f}^{(n_3^{'},n_4^{'},n_5-i_5+2,1,\cdots,1)}, \\ \vdots \\ \mathrm{conj}(\mathrm{A_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}) = \mathrm{A_f}^{(n_3^{'},n_4^{'},n_5^{'},\cdots,n_{d-1}^{'},n_d-i_d+2)}, \end{cases} \tag{22}$$

for $i_3 \in \left\{2, \cdots, \lceil \frac{n_3+1}{2} \rceil\right\}, \cdots, i_d \in \left\{2, \cdots, \lceil \frac{n_d+1}{2} \rceil\right\}$*;* $i_3^{'} \in \left\{1, \cdots, \lceil \frac{n_3+1}{2} \rceil\right\}, \cdots, i_{d-1}^{'} \in \left\{1, \cdots, \lceil \frac{n_{d-1}+1}{2} \rceil\right\}$*. Here, conj(·) is the conjugate operator. The explicit expressions of* $n_j^{'} (j = 3, 4, 5, \cdots, d)$ *can be written as follow:*

$$n_j^{'} = \begin{cases} n_j^{'} - i_j^{'} + 2, & i_j^{'} \neq 1 \\ 1, & i_j^{'} = 1 \end{cases}.$$

On the contrary, for any given $\mathcal{A}_\mathrm{f} \in \mathbb{C}^{n_1 \times \cdots \times n_d}$ *satisfying (22), there exists a real tensor* $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ *such that (21) holds. Leveraging on conjugate symmetry (22), the computational cost for order-*d *t-product, order-*d *t-SVD, order-*d *rt-SVD, order-*d *WTSN proximal operator and the robust low-rank tensor completion algorithm can be further reduced.*

Algorithm 11: FFT based order-d t-SVD, tsvd-fft($\boldsymbol{\mathcal{A}}$).

Input: $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_{d-1} \times n_d}$.
Output: t-SVD components $\boldsymbol{\mathcal{U}}$,$\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{V}}$ of $\boldsymbol{\mathcal{A}}$.

Compute the result of FFT on $\boldsymbol{\mathcal{A}}$
for $i = 3, 4, \cdots, d$ **do**
  $\boldsymbol{\mathcal{A}}_{\mathrm{f}} \leftarrow \mathrm{fft}(\boldsymbol{\mathcal{A}}, [\,], i)$;
end
Compute the matrix slice of $\boldsymbol{\mathcal{U}}_{\mathrm{f}}$, $\boldsymbol{\mathcal{S}}_{\mathrm{f}}$ and $\boldsymbol{\mathcal{V}}_{\mathrm{f}}$ from $\boldsymbol{\mathcal{A}}_{\mathrm{f}}$
for $i_3 \in \{2, \cdots, \lceil \frac{n_3+1}{2} \rceil\}, \cdots, i_d \in \{2, \cdots, \lceil \frac{n_d+1}{2} \rceil\}$,
$i_3^{'} \in \{1, \cdots, \lceil \frac{n_3+1}{2} \rceil\}, \cdots, i_{d-1}^{'} \in \{1, \cdots, \lceil \frac{n_{d-1}+1}{2} \rceil\}$ **do**

$$\begin{cases} \big[\mathrm{U_f}^{(1,1,1,\cdots,1,1)}, \mathrm{S_f}^{(1,1,1,\cdots,1,1)}, \mathrm{V_f}^{(1,1,1,\cdots,1,1)}\big] = \mathrm{svd}\big(\mathrm{A_f}^{(1,1,1,\cdots,1,1)}\big), \\ \big[\mathrm{U_f}^{(i_3,1,1,\cdots,1,1)}, \mathrm{S_f}^{(i_3,1,1,\cdots,1,1)}, \mathrm{V_f}^{(i_3,1,1,\cdots,1,1)}\big] = \mathrm{svd}\big(\mathrm{A_f}^{(i_3,1,1,\cdots,1,1)}\big), \\ \big[\mathrm{U_f}^{(i_3^{'},i_4,1,\cdots,1,1)}, \mathrm{S_f}^{(i_3^{'},i_4,1,\cdots,1,1)}, \mathrm{V_f}^{(i_3^{'},i_4,1,\cdots,1,1)}\big] = \mathrm{svd}\big(\mathrm{A_f}^{(i_3^{'},i_4,1,\cdots,1,1)}\big), \\ \big[\mathrm{U_f}^{(i_3^{'},i_4^{'},i_5,\cdots,1,1)}, \mathrm{S_f}^{(i_3^{'},i_4^{'},i_5,\cdots,1,1)}, \mathrm{V_f}^{(i_3^{'},i_4^{'},i_5,\cdots,1,1)}\big] = \mathrm{svd}\big(\mathrm{A_f}^{(i_3^{'},i_4^{'},i_5,\cdots,1,1)}\big), \\ \cdots\cdots \\ \big[\mathrm{U_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}, \mathrm{S_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}, \mathrm{V_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}\big] = \mathrm{svd}\big(\mathrm{A_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}\big); \end{cases}$$

end
Compute the remaining matrix slice of $\boldsymbol{\mathcal{U}}_{\mathrm{f}}$, $\boldsymbol{\mathcal{S}}_{\mathrm{f}}$ and $\boldsymbol{\mathcal{V}}_{\mathrm{f}}$ via (24)-(26);
Compute the result of inverse FFT on $\boldsymbol{\mathcal{U}}_{\mathrm{f}}$, $\boldsymbol{\mathcal{S}}_{\mathrm{f}}$ and $\boldsymbol{\mathcal{V}}_{\mathrm{f}}$
for $i = d, d-1, \cdots, 3$ **do**
  $\boldsymbol{\mathcal{U}} \leftarrow \mathrm{ifft}(\boldsymbol{\mathcal{U}}_{\mathrm{f}}, [\,], i), \boldsymbol{\mathcal{S}} \leftarrow \mathrm{ifft}(\boldsymbol{\mathcal{S}}_{\mathrm{f}}, [\,], i), \boldsymbol{\mathcal{V}} \leftarrow \mathrm{ifft}(\boldsymbol{\mathcal{V}}_{\mathrm{f}}, [\,], i)$.
end

Algorithm 12: FFT based order-d rt-SVD, rtsvd-fft($\boldsymbol{\mathcal{A}}, k, q, p$).

Input: $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_{d-1} \times n_d}$, truncation term $k < \min(n_1, n_2)$, oversampling parameter: $q > 0, p \geq 0$
Output: rt-SVD components $\boldsymbol{\mathcal{U}}$,$\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{V}}$ of $\boldsymbol{\mathcal{A}}$.

Set $l = k + q$ and initialize a Gaussian random tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{n_2 \times l \times n_3 \times \cdots \times n_d}$;
Compute the result of FFT on $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{G}}$
for $i = 3, 4, \cdots, d$ **do**
  $\boldsymbol{\mathcal{A}}_{\mathrm{f}} \leftarrow \mathrm{fft}(\boldsymbol{\mathcal{A}}, [\,], i), \boldsymbol{\mathcal{G}}_{\mathrm{f}} \leftarrow \mathrm{fft}(\boldsymbol{\mathcal{G}}, [\,], i)$;
end
Compute the matrix slice of $\boldsymbol{\mathcal{U}}_{\mathrm{f}}$, $\boldsymbol{\mathcal{S}}_{\mathrm{f}}$ and $\boldsymbol{\mathcal{V}}_{\mathrm{f}}$ from $\boldsymbol{\mathcal{A}}_{\mathrm{f}}$
for $i_3 \in \{2, \cdots, \lceil \frac{n_3+1}{2} \rceil\}, \cdots, i_d \in \{2, \cdots, \lceil \frac{n_d+1}{2} \rceil\}$,
$i_3^{'} \in \{1, \cdots, \lceil \frac{n_3+1}{2} \rceil\}, \cdots, i_{d-1}^{'} \in \{1, \cdots, \lceil \frac{n_{d-1}+1}{2} \rceil\}$ **do**

$$\begin{cases} \big[\mathrm{U_f}^{(1,1,1,\cdots,1,1)}, \mathrm{S_f}^{(1,1,1,\cdots,1,1)}, \mathrm{V_f}^{(1,1,1,\cdots,1,1)}\big] = \text{r-svd}\big(\mathrm{A_f}^{(1,1,1,\cdots,1,1)}, \mathrm{G_f}^{(1,1,1,\cdots,1,1)}, k, q, p\big), \\ \big[\mathrm{U_f}^{(i_3,1,1,\cdots,1,1)}, \mathrm{S_f}^{(i_3,1,1,\cdots,1,1)}, \mathrm{V_f}^{(i_3,1,1,\cdots,1,1)}\big] = \text{r-svd}\big(\mathrm{A_f}^{(i_3,1,1,\cdots,1,1)}, \mathrm{G_f}^{(i_3,1,1,\cdots,1,1)}, k, q, p\big), \\ \big[\mathrm{U_f}^{(i_3^{'},i_4,1,\cdots,1,1)}, \mathrm{S_f}^{(i_3^{'},i_4,1,\cdots,1,1)}, \mathrm{V_f}^{(i_3^{'},i_4,1,\cdots,1,1)}\big] = \text{r-svd}\big(\mathrm{A_f}^{(i_3^{'},i_4,1,\cdots,1,1)}, \mathrm{G_f}^{(i_3^{'},i_4,1,\cdots,1,1)}, k, q, p\big), \\ \big[\mathrm{U_f}^{(i_3^{'},i_4^{'},i_5,\cdots,1,1)}, \mathrm{S_f}^{(i_3^{'},i_4^{'},i_5,\cdots,1,1)}, \mathrm{V_f}^{(i_3^{'},i_4^{'},i_5,\cdots,1,1)}\big] = \text{r-svd}\big(\mathrm{A_f}^{(i_3^{'},i_4^{'},i_5,\cdots,1,1)}, \mathrm{G_f}^{(i_3^{'},i_4^{'},i_5,\cdots,1,1)}, k, q, p\big), \\ \cdots\cdots \\ \big[\mathrm{U_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}, \mathrm{S_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}, \mathrm{V_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}\big] = \text{r-svd}\big(\mathrm{A_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}, \\ \mathrm{G_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}, k, q, p\big); \end{cases}$$

end
Compute the remaining matrix slice of $\boldsymbol{\mathcal{U}}_{\mathrm{f}}$, $\boldsymbol{\mathcal{S}}_{\mathrm{f}}$ and $\boldsymbol{\mathcal{V}}_{\mathrm{f}}$ via (24)-(26);
Compute the result of inverse FFT on $\boldsymbol{\mathcal{U}}_{\mathrm{f}}$, $\boldsymbol{\mathcal{S}}_{\mathrm{f}}$ and $\boldsymbol{\mathcal{V}}_{\mathrm{f}}$
for $i = d, d-1, \cdots, 3$ **do**
  $\boldsymbol{\mathcal{U}} \leftarrow \mathrm{ifft}(\boldsymbol{\mathcal{U}}_{\mathrm{f}}, [\,], i), \boldsymbol{\mathcal{S}} \leftarrow \mathrm{ifft}(\boldsymbol{\mathcal{S}}_{\mathrm{f}}, [\,], i), \boldsymbol{\mathcal{V}} \leftarrow \mathrm{ifft}(\boldsymbol{\mathcal{V}}_{\mathrm{f}}, [\,], i)$.
end

Remark 4.2. *We let the SVD or randomized SVD (r-SVD) of* ${\mathrm{A}_{\mathrm{f}}}^{(i_3,i_4,\cdots,i_{d-1},i_d)}$ *be*

$$\begin{cases} \big[{\mathrm{U}_{\mathrm{f}}}^{(1,1,1,\cdots,1,1)}, {\mathrm{S}_{\mathrm{f}}}^{(1,1,1,\cdots,1,1)}, {\mathrm{V}_{\mathrm{f}}}^{(1,1,1,\cdots,1,1)}\big] = \Phi\big({\mathrm{A}_{\mathrm{f}}}^{(1,1,1,\cdots,1,1)}\big), \\ \big[{\mathrm{U}_{\mathrm{f}}}^{(i_3,1,1,\cdots,1,1)}, {\mathrm{S}_{\mathrm{f}}}^{(i_3,1,1,\cdots,1,1)}, {\mathrm{V}_{\mathrm{f}}}^{(i_3,1,1,\cdots,1,1)}\big] = \Phi\big({\mathrm{A}_{\mathrm{f}}}^{(i_3,1,1,\cdots,1,1)}\big), \\ \big[{\mathrm{U}_{\mathrm{f}}}^{(i_3',i_4,1,\cdots,1,1)}, {\mathrm{S}_{\mathrm{f}}}^{(i_3',i_4,1,\cdots,1,1)}, {\mathrm{V}_{\mathrm{f}}}^{(i_3',i_4,1,\cdots,1,1)}\big] = \Phi\big({\mathrm{A}_{\mathrm{f}}}^{(i_3',i_4,1,\cdots,1,1)}\big), \\ \big[{\mathrm{U}_{\mathrm{f}}}^{(i_3',i_4',i_5,\cdots,1,1)}, {\mathrm{S}_{\mathrm{f}}}^{(i_3',i_4',i_5,\cdots,1,1)}, {\mathrm{V}_{\mathrm{f}}}^{(i_3',i_4',i_5,\cdots,1,1)}\big] = \Phi\big({\mathrm{A}_{\mathrm{f}}}^{(i_3',i_4',i_5,\cdots,1,1)}\big), \\ \vdots \\ \big[{\mathrm{U}_{\mathrm{f}}}^{(i_3',i_4',i_5',\cdots,i_{d-1}',i_d)}, {\mathrm{S}_{\mathrm{f}}}^{(i_3',i_4',i_5',\cdots,i_{d-1}',i_d)}, {\mathrm{V}_{\mathrm{f}}}^{(i_3',i_4',i_5',\cdots,i_{d-1}',i_d)}\big] = \Phi\big({\mathrm{A}_{\mathrm{f}}}^{(i_3',i_4',i_5',\cdots,i_{d-1}',i_d)}\big), \end{cases} \tag{23}$$

for $i_3 \in \Big\{2,\cdots,\lceil\frac{n_3+1}{2}\rceil\Big\}$, $\cdots$, $i_d \in \Big\{2,\cdots,\lceil\frac{n_d+1}{2}\rceil\Big\}$, $i_3' \in \Big\{1,\cdots,\lceil\frac{n_3+1}{2}\rceil\Big\}$, $\cdots$, $i_{d-1}' \in \Big\{1,\cdots,\lceil\frac{n_{d-1}+1}{2}\rceil\Big\}$, *where* Φ *denotes decomposition operator. Here, the singular values in* ${\mathrm{S}_{\mathrm{f}}}^{(i_3,i_4,\cdots,i_d)}$ *are real. Besides, we let*

$$\begin{cases} {\mathrm{U}_{\mathrm{f}}}^{(1,1,1,\cdots,1,1)} \in \mathbb{R}^{n_1\times n_2}, \\ {\mathrm{U}_{\mathrm{f}}}^{(i_3,1,1,\cdots,1,1)} = \mathrm{conj}({\mathrm{U}_{\mathrm{f}}}^{(n_3-i_3+2,1,1,\cdots,1,1)}), \\ {\mathrm{U}_{\mathrm{f}}}^{(i_3,i_4,1,\cdots,1,1)} = \mathrm{conj}({\mathrm{U}_{\mathrm{f}}}^{(n_3',n_4-i_4+2,1,\cdots,1,1)}), \\ {\mathrm{U}_{\mathrm{f}}}^{(i_3,i_4,i_5,1,\cdots,1)} = \mathrm{conj}({\mathrm{U}_{\mathrm{f}}}^{(n_3',n_4',n_5-i_5+2,1,\cdots,1)}), \\ \vdots \\ {\mathrm{U}_{\mathrm{f}}}^{(i_3,i_4,i_5,\cdots,i_{d-1},i_d)} = \mathrm{conj}({\mathrm{U}_{\mathrm{f}}}^{(n_3',n_4',n_5',\cdots,n_{d-1}',n_d-i_d+2)}), \end{cases} \tag{24}$$

$$\begin{cases} {\mathrm{S}_{\mathrm{f}}}^{(1,1,1,\cdots,1,1,1)} \in \mathbb{R}^{n_1\times n_2}, \\ {\mathrm{S}_{\mathrm{f}}}^{(i_3,1,1,\cdots,1,1,1)} = ({\mathrm{S}_{\mathrm{f}}}^{(n_3-i_3+2,1,1,\cdots,1,1,1)}), \\ {\mathrm{S}_{\mathrm{f}}}^{(i_3,i_4,1,\cdots,1,1,1)} = ({\mathrm{S}_{\mathrm{f}}}^{(n_3',n_4-i_4+2,1,\cdots,1,1,1)}), \\ {\mathrm{S}_{\mathrm{f}}}^{(i_3,i_4,i_5,1,\cdots,1,1)} = ({\mathrm{S}_{\mathrm{f}}}^{(n_3',n_4',n_5-i_5+2,1,\cdots,1,1)}), \\ \vdots \\ {\mathrm{S}_{\mathrm{f}}}^{(i_3,i_4,i_5,\cdots,i_{d-2},i_{d-1},i_d)} = ({\mathrm{S}_{\mathrm{f}}}^{(n_3',n_4',n_5',\cdots,n_{d-2}',n_{d-1}',n_d-i_d+2)}), \end{cases} \tag{25}$$

$$\begin{cases} {\mathrm{V}_{\mathrm{f}}}^{(1,1,1,\cdots,1,1)} \in \mathbb{R}^{n_1\times n_2}, \\ {\mathrm{V}_{\mathrm{f}}}^{(i_3,1,1,\cdots,1,1)} = \mathrm{conj}({\mathrm{V}_{\mathrm{f}}}^{(n_3-i_3+2,1,1,\cdots,1,1)}), \\ {\mathrm{V}_{\mathrm{f}}}^{(i_3,i_4,1,\cdots,1,1)} = \mathrm{conj}({\mathrm{V}_{\mathrm{f}}}^{(n_3',n_4-i_4+2,1,\cdots,1,1)}), \\ {\mathrm{V}_{\mathrm{f}}}^{(i_3,i_4,i_5,1,\cdots,1)} = \mathrm{conj}({\mathrm{V}_{\mathrm{f}}}^{(n_3',n_4',n_5-i_5+2,1,\cdots,1)}), \\ \vdots \\ {\mathrm{V}_{\mathrm{f}}}^{(i_3,i_4,i_5,\cdots,i_{d-1},i_d)} = \mathrm{conj}({\mathrm{V}_{\mathrm{f}}}^{(n_3',n_4',n_5',\cdots,n_{d-1}',n_d-i_d+2)}), \end{cases} \tag{26}$$

for $i_3 \in \Big\{\lceil\frac{n_3+1}{2}\rceil+1,\cdots,n_3\Big\},\cdots,i_d \in \Big\{\lceil\frac{n_d+1}{2}\rceil+1,\cdots,n_d\Big\}$.

In virtue of (22)-(26), we present FFT based order-d t-product, FFT based order-d t-SVD, FFT based order-d rt-SVD, FFT based order-d t-QR, and FFT based order-d WTSN proximal operator in Algorithm 10-14, respectively.

5. REFERENCES

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

[2] M. E. Kilmer and C. D. Martin, "Factorization strategies for third-order tensors," *Linear Alg. Appl.*, vol. 435, no. 3, pp. 641–658, 2011.

[3] M. E. Kilmer, K. Braman, N. Hao, and R. C. Hoover, "Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging," *SIAM J. Matrix Anal. Appl.*, vol. 34, no. 1, pp. 148–172, 2013.

Algorithm 13: FFT based order-d t-QR decomposition, tqr-fft($\mathcal{A}$).

Input: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_{d-1} \times n_d}$.
Output: t-QR decomposition components $\mathcal{Q}$ and $\mathcal{R}$ of $\mathcal{A}$.

Compute the result of FFT on $\mathcal{A}$
for $i = 3, 4, \cdots, d$ **do**
  $\mathcal{A}_{\mathrm{f}} \leftarrow \mathrm{fft}(\mathcal{A}, [\,], i)$;
end
Compute the matrix slice of $\mathcal{Q}_{\mathrm{f}}$ and $\mathcal{R}_{\mathrm{f}}$ from $\mathcal{A}_{\mathrm{f}}$
for $i_3 \in \{2, \cdots, \lceil \frac{n_3+1}{2} \rceil\}, \cdots, i_d \in \{2, \cdots, \lceil \frac{n_d+1}{2} \rceil\}$,
$i_3^{'} \in \{1, \cdots, \lceil \frac{n_3+1}{2} \rceil\}, \cdots, i_{d-1}^{'} \in \{1, \cdots, \lceil \frac{n_{d-1}+1}{2} \rceil\}$ **do**

$$\begin{cases} \left[\mathrm{Q_f}^{(1,1,1,\cdots,1,1)}, \mathrm{R_f}^{(1,1,1,\cdots,1,1)}\right] = \mathrm{qr}\left(\mathrm{A_f}^{(1,1,1,\cdots,1,1)}, 0\right), \\ \left[\mathrm{Q_f}^{(i_3,1,1,\cdots,1,1)}, \mathrm{R_f}^{(i_3,1,1,\cdots,1,1)}\right] = \mathrm{qr}\left(\mathrm{A_f}^{(i_3,1,1,\cdots,1,1)}, 0\right), \\ \left[\mathrm{Q_f}^{(i_3^{'},i_4,1,\cdots,1,1)}, \mathrm{R_f}^{(i_3^{'},i_4,1,\cdots,1,1)}\right] = \mathrm{qr}\left(\mathrm{A_f}^{(i_3^{'},i_4,1,\cdots,1,1)}, 0\right), \\ \left[\mathrm{Q_f}^{(i_3^{'},i_4^{'},i_5,\cdots,1,1)}, \mathrm{R_f}^{(i_3^{'},i_4^{'},i_5,\cdots,1,1)}\right] = \mathrm{qr}\left(\mathrm{A_f}^{(i_3^{'},i_4^{'},i_5,\cdots,1,1)}, 0\right), \\ \cdots\cdots \\ \left[\mathrm{Q_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}, \mathrm{R_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}\right] = \mathrm{qr}\left(\mathrm{A_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}, 0\right); \end{cases}$$

end
Compute the remaining matrix slice of $\mathcal{Q}_{\mathrm{f}}$ and $\mathcal{R}_{\mathrm{f}}$ via by conjugate symmetry (22);
Compute the result of inverse FFT on $\mathcal{Q}_{\mathrm{f}}$ and $\mathcal{R}_{\mathrm{f}}$
for $i = d, d-1, \cdots, 3$ **do**
  $\mathcal{Q} \leftarrow \mathrm{ifft}(\mathcal{Q}_{\mathrm{f}}, [\,], i), \mathcal{R} \leftarrow \mathrm{ifft}(\mathcal{R}_{\mathrm{f}}, [\,], i)$.
end

Algorithm 14: FFT based order-d WTSN proximal operator

Input: $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, weight parameter: $\mathcal{W} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, truncation term: $k < \min(n_1, n_2)$, $\tau > 0$, $0 < p < 1$, the number of GST and PowerMethod iteration: J, I, oversampling parameter: $q > 0$.
Output: $\mathcal{D}_{\mathcal{W},p,\tau}(\mathcal{A}) = \mathcal{U} *_L \mathcal{S}_{\mathcal{W},p,\tau} *_L \mathcal{V}^*$.

Compute the rt-SVD or t-SVD components $\mathcal{U}$,$\mathcal{S}$ and $\mathcal{V}$ of $\mathcal{A}$
if *utilize the rt-SVD scheme* **then**
  $[\mathcal{U}, \mathcal{S}, \mathcal{V}] = \text{rtsvd-fft}(\mathcal{A}, k, q, I)$; `// rtsvd-fft(·) see Algorithm 12`
end
else if *utilize the t-SVD scheme* **then**
  $[\mathcal{U}, \mathcal{S}, \mathcal{V}] = \text{tsvd-fft}(\mathcal{A})$; `// tsvd-fft(·) see Algorithm 11`
end
Compute the matrix slice of $\mathcal{Z}_L$ for given index by
for $i_3 \in \{2, \cdots, \lceil \frac{n_3+1}{2} \rceil\}, \cdots, i_d \in \{2, \cdots, \lceil \frac{n_d+1}{2} \rceil\}$
$i_3^{'} \in \{1, \cdots, \lceil \frac{n_3+1}{2} \rceil\}, \cdots, i_{d-1}^{'} \in \{1, \cdots, \lceil \frac{n_{d-1}+1}{2} \rceil\}$ **do**

$$\begin{cases} \mathrm{Z_f}^{(1,1,1,\cdots,1,1)} = \mathrm{diag}\left\{\mathrm{GST}\left(\mathrm{diag}\,(\mathrm{S_f}^{(1,1,1,\cdots,1,1)}), \tau\,\mathrm{diag}\,(\mathrm{W}^{(1,1,1,\cdots,1,1)}), p, J\right)\right\}, \\ \mathrm{Z_f}^{(i_3,1,1,\cdots,1,1)} = \mathrm{diag}\left\{\mathrm{GST}\left(\mathrm{diag}\,(\mathrm{S_f}^{(i_3,1,1,\cdots,1,1)}), \tau\,\mathrm{diag}\,(\mathrm{W}^{(i_3,1,1,\cdots,1,1)}), p, J\right)\right\}, \\ \mathrm{Z_f}^{(i_3^{'},i_4,1,\cdots,1,1)} = \mathrm{diag}\left\{\mathrm{GST}\left(\mathrm{diag}\,(\mathrm{S_f}^{(i_3^{'},i_4,1,\cdots,1,1)}), \tau\,\mathrm{diag}\,(\mathrm{W}^{(i_3^{'},i_4,1,\cdots,1,1)}), p, J\right)\right\}, \\ \cdots\cdots \\ \mathrm{Z_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)} = \mathrm{diag}\left\{\mathrm{GST}\left(\mathrm{diag}\,(\mathrm{S_f}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}), \tau\,\mathrm{diag}\,(\mathrm{W}^{(i_3^{'},i_4^{'},i_5^{'},\cdots,i_{d-1}^{'},i_d)}), p, J\right)\right\}; \end{cases}$$

end
Compute the remaining matrix slice of $\mathcal{Z}_{\mathrm{f}}$ by conjugate symmetry (22);
Compute the result of inverse FFT on $\mathcal{Z}_{\mathrm{f}}$
for $i = d, d-1, \cdots, 3$ **do**
  $\mathcal{S}_{\mathcal{W},p,\tau} \leftarrow \mathrm{ifft}(\mathcal{Z}_{\mathrm{f}}, [\,], i)$;
end
Compute $\mathcal{D}_{\mathcal{W},p,\tau}(\mathcal{A}) = \text{tpro-fft}\left(\text{tpro-fft}(\mathcal{U}, \mathcal{S}_{\mathcal{W},p,\tau}), \mathcal{V}^*\right)$. `// tpro-fft(·) see Algorithm 10`

[4] C. D. Martin, R. Shafer, and B. LaRue, "An order-p tensor factorization with applications in imaging," *SIAM J. Sci. Comput.*, vol. 35, no. 1, pp. A474–A490, 2013.

[5] E. Kernfeld, M. Kilmer, and S. Aeron, "Tensor–tensor products with invertible linear transforms," *Linear Alg. Appl.*, vol. 485, pp. 545–570, 2015.

[6] W. Qin, H. Wang, F. Zhang, M. Dai, and J. Wang, "Robust low-rank tensor reconstruction using high-order t-SVD," *Journal of Electronic Imaging*, vol. 30, no. 6, p. 063016, 2021.

[7] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Rev.*, vol. 53, no. 2, pp. 217–288, 2011.

[8] Q. Yao and J. T. Kwok, "Accelerated and inexact soft-impute for large-scale matrix and tensor completion," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 9, pp. 1665–1679, 2018.

[9] W. Zuo, D. Meng, L. Zhang, X. Feng, and D. Zhang, "A generalized iterated shrinkage algorithm for non-convex sparse coding," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2013, pp. 217–224.

[10] Y. Xie, S. Gu, Y. Liu, W. Zuo, W. Zhang, and L. Zhang, "Weighted schatten p-norm minimization for image denoising and background subtraction," *IEEE Trans. Image Process.*, vol. 25, no. 10, pp. 4842–4857, 2016.

[11] O. Rojo and H. Rojo, "Some results on symmetric circulant matrices and on symmetric centrosymmetric matrices," *Linear algebra and its applications*, vol. 392, pp. 211–233, 2004.