Computational and Mathematical Methods

Research Article

C-Product Toolbox: A Computational Package for Third-Order Tensor Operations Based on the Reduced C-Product

Submission ID aa907694-e48a-4500-bd6e-745670926c29

Submission Version Revision 1

PDF Generation 24 Jul 2025 18:05:05 EST by Atypon ReX

Authors

Dr. Pablo Soto-Quiros
Corresponding Author
Submitting Author

Affiliations

- Escuela de Matemática, Instituto Tecnológico de Costa Rica, Cartago 30101, Costa Rica
- Escuela de Ingeniería en Computadores, Instituto Tecnológico de Costa Rica, Cartago 30101, Costa Rica

Prof. Samuel Valverde-Sanchez

Affiliations

- Escuela de Matemática, Instituto Tecnológico de Costa Rica, Cartago 30101, Costa Rica

Prof. Luis Chavarria-Zamora

Affiliations

- Escuela de Ingeniería en Computadores, Instituto Tecnológico de Costa Rica, Cartago 30101, Costa Rica

Additional Information

Files for peer review

All files submitted by the author for peer review are listed below. Files that could not be converted to PDF are indicated; reviewers are able to access them online.

Name	Type of File	Size	Page
author_response_R2.pdf	Author Response	74.5 KB	Page 3
template_paper.pdf	Main Document - LaTeX PDF	5.7 MB	Page 4

July 24, 2025

Dr. Jesús Vigo Aguiar
Chief Editor
Computational and Mathematical Methods

Dear Editor,

Good morning. I hope this message finds you well.

Below, I provide responses to the comments presented by Reviewer 1 for the paper *C-Product Toolbox: A computational package for third-order tensor operations based on the reduced c-product*, (Manuscript ID: 6048327)

1. **Comment:** There is a grammatical issue. For example, in line 5 on Page 14 of the main text, "an therefore $B = A^{-1}$" should be corrected to "and therefore $B = A^{-1}$".
   Answer: Thank you very much for your observation. The grammatical issue has been corrected.
2. **Comment:** Capitalize the first letter of each major word in the title. For example, change the title on Page 1, "A computational package for third-order tensor operations based on the reduced c-product", to "A Computational Package for Third-Order Tensor Operations Based on the Reduced C-Product".
   Answer: Thank you for the suggestion. In the revised version of the paper, we have capitalized the first letter of each major word in the title, as requested.
3. **Comment:** Capitalize the first letter of each major word in all first-level headings. For example, change "2. Mathematical foundations of the C-Product Toolbox" on Page 4 to "2. Mathematical Foundations of the C-Product Toolbox".
   Answer: Thank you for the suggestion. In the revised version of the paper, we have capitalized the first letter of each first-level headings, as requested.
4. **Comment:** In Table 1 on Page 8, the headers "Tensor Product", "MATLAB code for computing $B = L(A)$", "Python code for computing $B = L(A)$", and "Arithmetic used in $L(A)$" do not capitalize the first letter of every major word. However, in Table 5 on Page 24, the headers "Test Case" and "Percent Difference" capitalize every word. We recommend unifying the capitalization style for all table headers throughout the paper.
   Answer: Thank you for your comment. We have now unified the capitalization style of all table headers throughout the paper, using sentence case (i.e., capitalizing only the first letter of the first word). An exception was made in Table 3, where the names of the computational packages — "*C-Product Toolbox*" and "*Tensor-Tensor Product Toolbox 2.0*" — retain capitalization of each major word, as they are proper names of the respective computational package.

Please let me know if anything else is required.

Best regards

Pablo Soto-Quiros, Samuel Valverde-Sanchez and Luis Chavarria-Zamora

C-Product Toolbox: A Computational Package for Third-Order Tensor Operations Based on the Reduced C-Product

Pablo Soto-Quiros[a,b], Samuel Valverde-Sanchez[a], Luis Chavarria-Zamora[b]

[a] *Escuela de Matemática, Instituto Tecnológico de Costa Rica, Cartago 30101, Costa Rica*
[b] *Escuela de Ingeniería en Computadores, Instituto Tecnológico de Costa Rica, Cartago 30101, Costa Rica*

Abstract

This paper introduces the *C-Product Toolbox*, a new computational package available for MATLAB and Python, designed to perform operations on third-order tensors using a tensor product known as the reduced c-product. The reduced c-product is a variant of the known c-product, a tensor product based on the discrete cosine transform and belonging to a family of tensor products defined by invertible linear transformations. This work presents the theoretical development of the reduced c-product used and provides a detailed explanation of each tensor operation implemented in the computational package. Additionally, numerical experiments compare the *C-Product Toolbox* with an existing MATLAB toolbox that employs the t-product, a tensor product based on the discrete Fourier transform. The numerical experiments in this paper demonstrate that the *C-Product Toolbox* offers superior computational efficiency, achieving faster execution times and lower memory consumption. Furthermore, the practical advantages of the proposed methods are highlighted through an application in video denoising, showcasing the effectiveness of the toolbox in real-world scenarios.

Keywords: Tensor Product, Linear Transformation, Discrete Cosine Transform

1. Introduction

A tensor $\mathcal{A} \in \mathbb{R}^{m_1 \times \cdots \times m_p}$ is a multidimensional array of numbers of order p, where $\mathcal{A}(i_1, \ldots, i_p) \in \mathbb{R}$ with $i_k = 1, \ldots, m_k$, for all $k = 1, \ldots, p$. Tensors generalize the concept of vectors and matrices. In particular, a third-order tensor $\mathcal{A} \in \mathbb{R}^{m \times n \times p}$ is a three-dimensional representation of a matrix, as shown in Figure 1(a).

Third-order tensors are widely used in various applications, including data clustering [1, 2, 3], machine learning [4, 5, 6], and image processing [7, 8, 9, 10, 11] . Figure 1(b) shows a color image in RGB format, where RGB stands for Red, Green, and Blue. This type of image can be naturally represented as a third-order tensor of size $m \times n \times 3$, where each element represents light intensity, and each frontal slice corresponds to a color channel (red,

Email addresses: jusoto@tec.ac.cr (Pablo Soto-Quiros), savalverde@tec.ac.cr (Samuel Valverde-Sanchez), lachavarria@itcr.ac.cr (Luis Chavarria-Zamora)

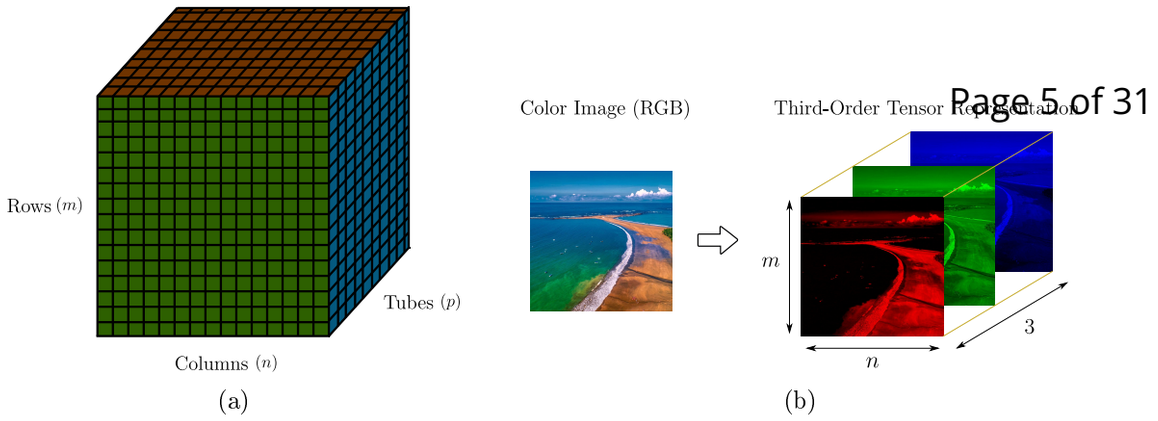


Figure 1: (a) Graphical representation of a third-order tensor $\mathcal{A} \in \mathbb{R}^{m \times n \times p}$. (b) Representation of an RGB image as a third-order tensor of size $m \times n \times 3$.

green, and blue, respectively). This paper focuses on third-order tensors, and thus the term *tensor* refer to these throughout this paper.

In general, within tensor algebra, the addition and subtraction of tensors are analogous to matrix addition and subtraction. That is, if $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{m \times n \times p}$, then $\mathcal{C}_1 = \mathcal{A} + \mathcal{B} \in \mathbb{R}^{m \times n \times p}$ and $\mathcal{C}_2 = \mathcal{A} - \mathcal{B} \in \mathbb{R}^{m \times n \times p}$, where $\mathcal{C}_1(i,j,k) = \mathcal{A}(i,j,k) + \mathcal{B}(i,j,k)$ and $\mathcal{C}_2(i,j,k) = \mathcal{A}(i,j,k) - \mathcal{B}(i,j,k)$, for all $i = 1, \ldots, m$, $j = 1, \ldots, n$, and $k = 1, \ldots, p$. However, different approaches to tensor multiplication exist, using various types of products, such as the Einstein product [12], the t-product [13], and the c-product [14]. The latter two are part of a family of tensor products involving invertible linear transformations known as the m-product [14]. The linear transformation related to the t-product relies on the discrete Fourier transform (DFT), and the one related to the c-product employs the discrete cosine transform (DCT).

The t-product, developed in [13], has been cited in over 1150 scientific papers according to Google Scholar. On the other hand, the c-product, introduced in [14], has received more than 290 citations in the same database. This suggests that the t-product is more commonly used than the c-product. This is because the t-product uses the benefits of the fast Fourier transform for its computation. However, as noted in [15], the t-product requires intermediate complex arithmetic, making it more computationally expensive. On the other hand, although the c-product benefits from real arithmetic, it can be computationally slower in certain cases. This is because, unlike the t-product, the c-product lacks a fast algorithm for its computation. It relies not only on the DCT but also requires the use of additional matrices for the calculation, as explained in [14].

Additionally, a computational package for third-order tensors based on products with invertible linear transformations was developed in MATLAB. This toolbox, named the *Tensor-Tensor Product Toolbox 2.0* [16, 17], facilitates tensor operations using products with invertible linear transformations. This toolbox defaults to the t-product, meaning that users must manually specify other linear transformations to define the desired tensor product each time a function is used, which can be a tedious process. It is also worth noting that the *Tensor-Tensor Product Toolbox 2.0* is only available for MATLAB, limiting its use in more common

programming environments like Python, and it has not been updated since 2021, making it less suitable for recent developments in tensor products.

Given these limitations and opportunities for improvement, this research develops a new computational package available for MATLAB (R2024b) and Python (3.12) called the *C-Product Toolbox*, specifically designed for tensor operations based on a variation of the c-product known as the *reduced c-product*. In the MATLAB implementation, version R2024b was used, while the Python version relies on the NumPy (v1.26.4) and SciPy (v1.16.0) libraries. The reduced c-product enables a more efficient computation of the tensor product based on the DCT (see Section 2.1 for more details), and has already been applied in previous studies, such as in [18] for the problem of low-rank tensor completion. The toolbox is available on GitHub at the following link:

https://github.com/jusotoTEC/c-product-toolbox

To ensure reproducibility, the specific version of the toolbox used in this paper, identified as version `v1.0.0`, is permanently archived and can be accessed at:

https://github.com/jusotoTEC/c-product-toolbox/releases/tag/v1.0.0

The primary motivation for developing the *C-Product Toolbox* stems from research related to signal and image processing, which uses the reduced c-product as the tensor product. While the *Tensor-Tensor Product Toolbox 2.0* served as inspiration, it no longer fully meets the needs of this research. This is because the t-product can demand more storage and computational resources, whereas the reduced c-product offers optimized performance in both areas. These improvements are demonstrated through numerical experiments presented in Section 4. Furthermore, the lack of recent updates in the *Tensor-Tensor Product Toolbox 2.0* limits its usefulness in light of current research advances.

In addition, this paper also include a review of key mathematical concepts related to tensors to enhance the robustness of the work. We first introduce the m-product, a generalization of tensor products under linear transformations, which serve as the foundation for defining the reduced c-product. We also highlight the differences between the reduced c-product and other tensor products developed within this framework. Additionally, this work offers a concise and detailed mathematical survey of the key operations and functions related to tensors using the reduced c-product. This advantage provides readers with a clear understanding of the theoretical foundations of these operations, facilitating their application in various contexts. By presenting essential concepts systematically, the review serves as a valuable reference for researchers and practitioners, enhancing the accessibility of complex ideas and promoting broader engagement with the *C-Product toolbox*.

Finally, this paper presents numerical experiments that highlight the advantages of the *C-Product Toolbox*, demonstrating its reduced computation time, significant acceleration, and overall improvement compared to the *Tensor-Tensor Product Toolbox 2.0*. Moreover, we provide a practical example showcasing how the toolbox can be used for video denoising.

In summary, the main contributions and advantages of this paper are as follows:

- The development of the *C-Product Toolbox*, a new computational package for third-order tensor operations based on the reduced c-product. Unlike the *Tensor-Tensor Product Toolbox 2.0*, which relies on the DFT and complex arithmetic, the *C-Product Toolbox* employs the DCT and only requires real arithmetic, resulting in lower memory consumption, reduced execution time, and improved computational efficiency.
- A review of key mathematical concepts related to tensors and the reduced c-product, providing a theoretical foundation for computational implementation in the *C-Product Toolbox*.
- The *C-Product Toolbox* includes operations not available in the *Tensor-Tensor Product Toolbox 2.0*, such as tensor pseudoinverse, Drazin tensor inverse, and least-squares solutions for tensor equations.
- The package has been implemented in both MATLAB and Python (using the `NumPy` and `SciPy` libraries), ensuring broader accessibility for scientific computing and applications in signal and image processing.
- A comprehensive GitHub repository provides the toolbox along with a user manual in English and Spanish, detailing installation and usage instructions.
- Numerical experiments demonstrate the advantages of the *C-Product Toolbox* over existing methods, highlighting its efficiency in tensor computations. Additionally, an application in video denoising showcases its practical utility.

The rest of this document is organized as follows. Section 2 presents a survey of the mathematical foundations of each function developed in the toolbox. Section 3 describes the functions included in the *C-Product Toolbox*, along with a brief explanation of their computational usage. Section 4 presents several numerical experiments that demonstrate the advantages of the *C-Product Toolbox*. Finally, Section 5 offers the concluding remarks.

2. Mathematical Foundations of the *C-Product Toolbox*

This section presents the mathematical theory underlying each function in the *C-Product Toolbox*. First, we introduce the key concepts essential for understanding this part of the study.

A third-order tensor $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ is a three-dimensional array of numbers, such that $\mathcal{A}(i,j,k) \in \mathbb{R}$, for all $i = 1,\dots,m$, $j = 1,\dots,n$, and $k = 1,\dots,p$. Using MATLAB-style indexing notation, we define the following partitions of a tensor:

- The *lateral slices* $\mathcal{A}(:,j,:) \in \mathbb{R}^{m\times 1\times p}$ are matrices aligned along the second dimension.
- The *frontal slices* $\mathcal{A}(:,:,k) \in \mathbb{R}^{m\times n}$ are stacked matrices along the third dimension. To simplify the notation, the k-th frontal slice of the tensor $\mathcal{A}$ is denoted as $\mathcal{A}^{(k)} = \mathcal{A}(:,:,k)$.

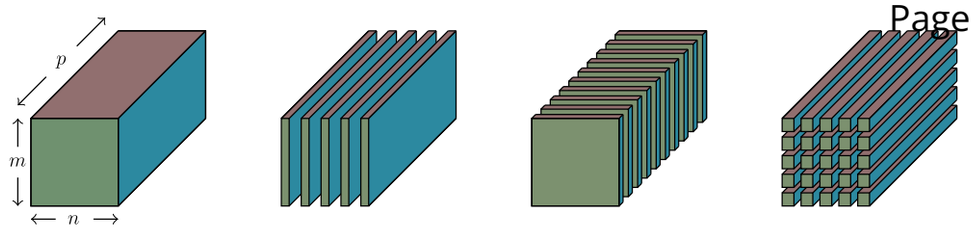


Figure 2: Graphical representation of the partitioning of a third-order tensor. (a) Tensor $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$. (b) Lateral slices $\mathcal{A}(:,j,:) \in \mathbb{R}^{m\times 1\times p}$. (c) Frontal slices $\mathcal{A}(:,:,k) \in \mathbb{R}^{m\times n}$. (d) Tubes $\mathcal{A}(i,j,:) \in \mathbb{R}^{1\times 1\times p}$.

- The *tubes* $\mathcal{A}(i,j,:) \in \mathbb{R}^{1\times 1\times p}$, also known as tubes fibers, are vectors spanning the third dimension.

Figure 2 provides a graphical illustration of these tensor partitions to facilitate comprehension.

2.1. The reduced c-product

Before formally defining the reduced c-product, we introduce some fundamental concepts that provide the necessary foundation for understanding them in this paper.

Definition 1 ([14]). *Let $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ and $\mathcal{B} \in \mathbb{R}^{n\times s\times p}$. The facewise product $\mathcal{C} = \mathcal{A} \triangle \mathcal{B} \in \mathbb{R}^{m\times s\times p}$ is defined as:*

$$\mathcal{C}^{(i)} = (\mathcal{A} \triangle \mathcal{B})^{(i)} = \mathcal{A}^{(i)}\mathcal{B}^{(i)},$$

where $\mathcal{A}^{(i)}$ and $\mathcal{B}^{(i)}$ denote the i-th frontal slices of tensors $\mathcal{A}$ and $\mathcal{B}$, respectively, for all $i = 1, ..., p$.

Definition 2 ([14, 19]). *The mode-3 product between a tensor $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ and a matrix $M \in \mathbb{R}^{q\times p}$ is the tensor $\mathcal{B} \in \mathbb{R}^{m\times n\times q}$ defined as $\mathcal{B} = \mathcal{A} \times_3 M$, where each element of $\mathcal{B}$ satisfies*

$$\mathcal{B}(i,j,:) = vec^{-1}(M\,vec(A(i,j,:))),$$

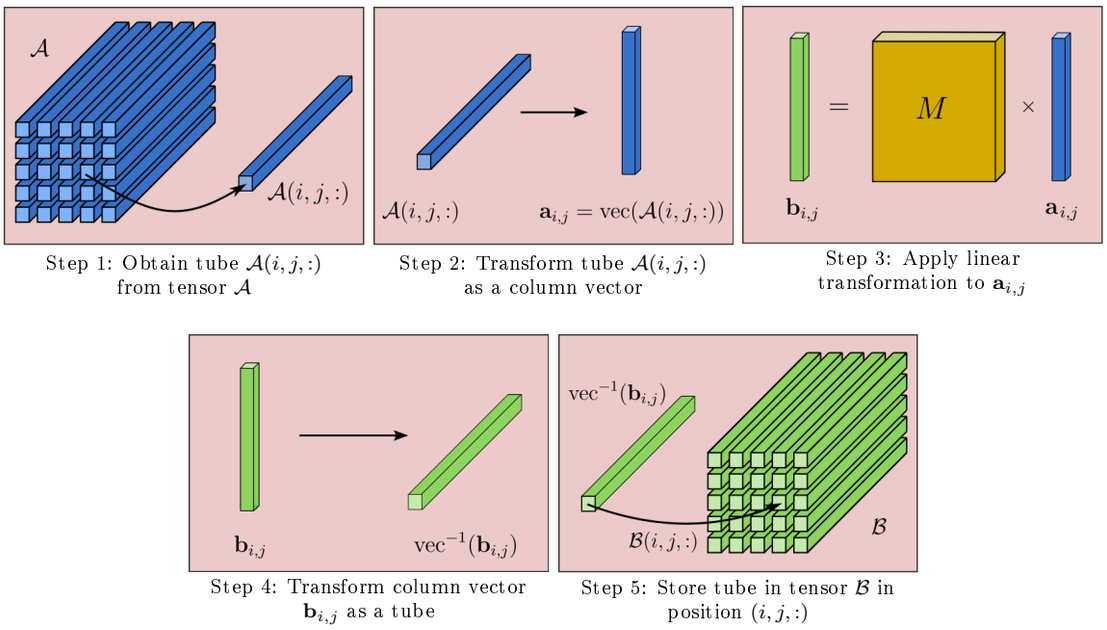
for all $i = 1, ..., m$ and $j = 1, ..., n$, with $vec : \mathbb{R}^{1\times 1\times p} \to \mathbb{R}^p$ being the operator that reshapes a tube vector into a column vector, and $vec^{-1} : \mathbb{R}^p \to \mathbb{R}^{1\times 1\times p}$ being its inverse.

The mode-3 product is an essential mathematical operation in tensor theory. Figure 3 graphically represents each of the steps required to perform the mode-3 product between a tensor and a matrix.

Definition 3 ([14]). *The m-product between two tensors $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ and $\mathcal{B} \in \mathbb{R}^{n\times s\times p}$ is the new tensor $\mathcal{A} *_M \mathcal{B} \in \mathbb{R}^{m\times s\times p}$ defined as*

$$\mathcal{A} *_M \mathcal{B} = L^{-1}(L(\mathcal{A}) \triangle L(\mathcal{B})), \tag{1}$$

where the linear operator L and its inverse L^{-1} are defined as $L(\mathcal{A}) = \mathcal{A}\times_3 M$ and $L^{-1}(\mathcal{A}) = \mathcal{A} \times_3 M^{-1}$. Here, $M \in \mathbb{R}^{p\times p}$ is a nonsingular matrix.



Step 1: Obtain tube $\mathcal{A}(i,j,:)$ from tensor $\mathcal{A}$

Step 2: Transform tube $\mathcal{A}(i,j,:)$ as a column vector

Step 3: Apply linear transformation to $\mathbf{a}_{i,j}$

Step 4: Transform column vector $\mathbf{b}_{i,j}$ as a tube

Step 5: Store tube in tensor $\mathcal{B}$ in position $(i,j,:)$

Figure 3: Graphical representation of the steps for performing the mode-3 product $\mathcal{B} = \mathcal{A} \times_3 M$.

There are two particular cases of the m-product that are commonly applied in tensor research. The first is the t-product, as defined in [13], where the linear transformation associated with the t-product relies on the DFT. The matrix associated with the operator L is given by $M = F_p$, where F_p is the DFT matrix with entries $F_p(i,j) = e^{-2\pi i(i-1)(j-1)/p}$, for $i,j = 1,...,p$. Additionally, using MATLAB commands `fft` and `ifft`, we have $L(\mathcal{A}) =$ `fft`$(\mathcal{A},[\,],3)$ and $L^{-1}(\mathcal{A}) =$ `ifft`$(\mathcal{A},[\,],3)$ for any third-order tensor $\mathcal{A}$. This enables the acceleration of the computation of $L(\mathcal{A})$ and $L^{-1}(\mathcal{A})$.

The second widely-used m-product case is the c-product, as introduced in [14]. Here, the matrix associated with the operator L is given by

$$M = W^{-1}C_p(I_p + Z),$$

where $C_p \in \mathbb{R}^{p\times p}$ corresponds to the type-II DCT matrix, defined as

$$C_p(i,j) = \sqrt{\frac{2}{p(1+\delta(i,1))}}\cos\left(\frac{\pi(2j-1)(i-1)}{2p}\right), \tag{2}$$

for $i,j = 1,2,...,p$, $W = \mathrm{diag}(C_p(:,1)) \in \mathbb{R}^{p\times p}$, $I_p \in \mathbb{R}^{p\times p}$ is the identity matrix, and $Z \in \mathbb{R}^{p\times p}$ is the upward-shift circulant matrix, defined as $Z = \mathrm{diag}(\mathrm{ones}(p-1,1),1)$. In (2), δ is the Kronecker delta function, defined by

$$\delta(i,j) = \begin{cases} 0, & \text{si } i \neq j \\ 1, & \text{si } i = j \end{cases}.$$

An advantage of the t-product is that it allows for efficient computation using the Fast Fourier Transform (FFT). However, as noted in [15], the t-product involves intermediate complex arithmetic, which increases computational expense compared to the real arithmetic operations employed by the c-product. Moreover, using complex arithmetic requires more memory than real arithmetic because each complex number stores both real and imaginary parts, doubling the storage needs and adding computational overhead.

On the other hand, the c-product, as proposed in [14], lacks dedicated fast computational commands in MATLAB or Python, unlike the t-product, which benefits from optimized commands such as `fft` and `ifft` for its computation.

Given the limitations previously discussed regarding the t-product and c-product, this paper introduces a simple yet effective variant of the c-product. Specifically, we consider the m-product defined in Definition 3, where we set $M = C_p$, with C_p being the type-II DCT matrix defined in (2), which is an orthogonal matrix. Therefore, we have $L(\mathcal{A}) = \mathcal{A} \times_3 C_p$ and $L^{-1}(\mathcal{A}) = \mathcal{A} \times_3 C_p^T$. This newly defined tensor product is referred to as the *reduced c-product.* To provide a comprehensive understanding, we present the following definition of the reduced c-product.

Definition 4 (Reduced c-product). *The reduced c-product between tensors $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ and $\mathcal{B} \in \mathbb{R}^{n\times s\times p}$ is the new tensor $\mathcal{A} *_c \mathcal{B} \in \mathbb{R}^{m\times s\times p}$ defined as*

$$\mathcal{A} *_c \mathcal{B} = L^{-1}(L(\mathcal{A}) \triangle L(\mathcal{B})), \tag{3}$$

where $L(\mathcal{A}) = \mathcal{A} \times_3 C_p$ and $L^{-1}(\mathcal{A}) = \mathcal{A} \times_3 C_p^T$.

The reduced c-product has already been applied in previous studies, such as in [18] for the problem of low-rank tensor completion.

Example 1. *Consider the tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{2\times 2\times 3}$, , where their frontal slices are defined as follows*

$$\mathcal{A}^{(1)} = \begin{pmatrix} -2 & 0 \\ 1 & 0 \end{pmatrix}, \quad \mathcal{A}^{(2)} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}, \quad \mathcal{A}^{(3)} = \begin{pmatrix} 3 & -2 \\ 0 & -1 \end{pmatrix}$$

and

$$\mathcal{B}^{(1)} = \begin{pmatrix} 2 & -1 \\ 0 & 0 \end{pmatrix}, \quad \mathcal{B}^{(2)} = \begin{pmatrix} -1 & -3 \\ 0 & 2 \end{pmatrix}, \quad \mathcal{B}^{(3)} = \begin{pmatrix} 1 & 2 \\ -1 & -1 \end{pmatrix}.$$

Applying Algorithm 1, we compute the tensor product $\mathcal{A} *_c \mathcal{B} = \mathcal{C} \in \mathbb{R}^{2\times 2\times 3}$, where the frontal slices of $\mathcal{C}$ are given by

$$\mathcal{C}^{(1)} = \begin{pmatrix} -0.462 & 4.595 \\ 1.284 & -0.059 \end{pmatrix}, \ \mathcal{C}^{(2)} = \begin{pmatrix} 2.844 & -1.211 \\ 0.577 & -3.026 \end{pmatrix}, \ \mathcal{C}^{(3)} = \begin{pmatrix} 1.659 & -7.425 \\ -0.129 & 1.354 \end{pmatrix}.$$

The reduced c-product introduced in Definition 4 enables the direct utilization of the MATLAB and Python commands `dct` and `idct` for calculations. In MATLAB, these transformations are represented as $L(\mathcal{A}) = \texttt{dct}(\mathcal{A}, [\,], 3)$ and $L^{-1}(\mathcal{A}) = \texttt{idct}(\mathcal{A}, [\,], 3)$, for any third-order tensor $\mathcal{A}$. Furthermore, this approach minimizes memory usage during tensor

Algorithm 1: Reduced c-product

Input : $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ y $\mathcal{B} \in \mathbb{R}^{n\times s\times p}$

Output: $\mathcal{C} = \mathcal{A} *_c \mathcal{B} \in \mathbb{R}^{m\times s\times p}$

$\overline{\mathcal{A}} = \texttt{dct}(\mathcal{A}, [\,], 3)$

$\overline{\mathcal{B}} = \texttt{dct}(\mathcal{B}, [\,], 3)$

$\overline{\mathcal{C}} = \texttt{zeros}(m, s, p)$

for $i = 1 : p$ **do**

  $\overline{\mathcal{C}}^{(i)} = \overline{\mathcal{A}}^{(i)}\overline{\mathcal{B}}^{(i)}$

$\mathcal{C} = \texttt{idct}(\overline{\mathcal{C}}, [\,], 3)$

Tensor product	**Matrix M in $L(\mathcal{A})$**	**MATLAB code for computing $\mathcal{B} = L(\mathcal{A})$**	**Python code for computing $\mathcal{B} = L(\mathcal{A})$**	**Arithmetic used in $L(\mathcal{A})$**
t-product	F_p	`B=fft(A,[],3)`	`B=fft(A,axis=0,norm='ortho')`	Complex

c-product	$W^{-1}C_p(I_p + Z)$	`Cp = dctmtx(p);`
		`W = diag(Cp(:,1));`
		`Z = diag(ones(p-1,1),1);`
		`M = W\(Cp*(eye(p)+Z));`
		`A_rsh = reshape(A,[],p);`
		`B_rsh = A_rsh*M';`
		`B = reshape(B_rsh,m,n,p);`
			`Ip = np.eye(p)`
			`Cp = sp.fftpack.dct(Ip,norm='ortho')`
			`W = np.diag(Cp[:,0])`
			`Z = np.diag(np.ones(p-1), 1)`
			`M = np.linalg.inv(W)@(Cp@(Ip+Z))`
			`A_rsh = A.reshape(-1,p)`
			`B_rsh = A_rsh@M.T`
			`B = B_rsh.reshape(m,n,p)`
				Real

reduced c-product	C_p	`B=dct(A,[],3)`	`B=dct(A,axis=0,norm='ortho')`	Real

Table 1: Summary on the t-product, the c-product, and the reduced c-product.

product computations and reduces execution times. These enhancements are illustrated through numerical experiments presented in Section 4. Algorithm 1 provides an efficient method for calculating the reduced c-product of tensors $\mathcal{A}$ and $\mathcal{B}$, using MATLAB commands `dct` and `idct`.

Finally, Table 1 summarizes the key features of the t-product, the c-product, and the reduced c-product discussed earlier, including their respective implementations in MATLAB and Python.

2.2. Tensor inner product, tensor identity, and tensor transpose

In this section, we introduce three miscellaneous tensor-related concepts that serve as the foundation for definitions used later in this document.

Definition 5 ([20]). *The inner product between two tensors $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ and $\mathcal{B} \in \mathbb{R}^{m\times n\times p}$ is defined as*

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i=1}^{p} tr((\mathcal{A}^{(i)})^T \mathcal{B}^{(i)}),$$

where $tr(C)$ denotes the trace of a square matrix C.

Example 2. *Consider the tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{2\times 2\times 3}$ from Example 1. The inner product of $\mathcal{A}$ and $\mathcal{B}$ is given by*

$$\begin{aligned}\langle \mathcal{A}, \mathcal{B}\rangle &= \sum_{i=1}^{3} tr((\mathcal{A}^{(i)})^T \mathcal{B}^{(i)}) \\ &= tr((\mathcal{A}^{(1)})^T \mathcal{B}^{(1)}) + tr((\mathcal{A}^{(2)})^T \mathcal{B}^{(2)}) + tr((\mathcal{A}^{(3)})^T \mathcal{B}^{(3)}) \\ &= -4 + -1 + 0 \\ &= -5\end{aligned}$$

Definition 6 ([14]). *The tensor $\mathcal{I}_{n,p} \in \mathbb{R}^{n\times n\times p}$ is called the identity tensor if*

$$\mathcal{A} *_c \mathcal{I}_{n,p} = \mathcal{I}_{n,p} *_c \mathcal{A} = \mathcal{A}.$$

Based on Definition 6, if $\overline{\mathcal{I}}_{n,p} = L(\mathcal{I}_{n,p})$, then the identity tensor satisfies $\overline{\mathcal{I}}_{n,p}^{(i)} = I_n$ for all $i = 1, \ldots, p$.

Example 3. *The tensor identity in $\mathbb{R}^{2\times 2\times 2}$ is denoted by $\mathcal{I}_{2,2}$, where its frontal slices are given by*

$$\mathcal{I}_{2,2}^{(1)} = \begin{pmatrix} \sqrt{2} & 0 \\ 0 & \sqrt{2} \end{pmatrix}, \quad \mathcal{I}_{2,2}^{(2)} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

This holds because, if we denote $\overline{\mathcal{I}}_{2,2} = L(\mathcal{I}_{2,2})$, then $\overline{\mathcal{I}}_{2,2}^{(1)} = \overline{\mathcal{I}}_{2,2}^{(2)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

Definition 7 ([14]). *The tensor transpose of $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ is the tensor $\mathcal{A}^T \in \mathbb{R}^{n\times m\times p}$, defined by applying the rule $L(\mathcal{A}^T)^{(i)} = L(\mathcal{A}^{(i)})^T$ for all $i = 1, \ldots, p$.*

Example 4. *The tensor transpose of $\mathcal{A} \in \mathbb{R}^{2\times 3\times 2}$, whose frontal slices are given by*

$$\mathcal{A}^{(1)} = \begin{pmatrix} 1 & -1 & -2 \\ -2 & 3 & 1 \end{pmatrix}, \quad \mathcal{A}^{(2)} = \begin{pmatrix} -2 & 3 & 1 \\ 1 & 0 & -1 \end{pmatrix},$$

is the tensor $\mathcal{B} \in \mathbb{R}^{3\times 2\times 2}$, whose frontal slices are defined as

$$\mathcal{B}^{(1)} = \begin{pmatrix} 1 & -2 \\ -1 & 3 \\ -2 & 1 \end{pmatrix}, \quad \mathcal{B}^{(2)} = \begin{pmatrix} -2 & 1 \\ 3 & 0 \\ 1 & -1 \end{pmatrix},$$

where $\mathcal{B} = \mathcal{A}^T$.

2.3. Tensor factorization methods

In this section, we present definitions for three important tensor factorizations: *the c-SVD*, *the c-QR*, and *the tensor square root.*

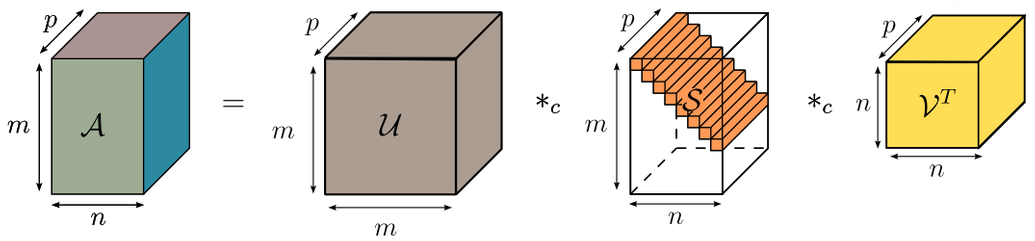


Figure 4: Graphic representation of the c-SVD of $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$.

2.3.1. c-SVD

The c-SVD is the tensor singular value decomposition (SVD) under the reduced c-product. Before defining the c-SVD, we introduce some preliminary concepts as follows.

- A tensor $\mathcal{Q} \in \mathbb{R}^{m\times m\times p}$ is called an orthogonal tensor [14] if it satisfies the property

$$\mathcal{Q} *_c \mathcal{Q}^T = \mathcal{Q}^T *_c \mathcal{Q} = \mathcal{I}_{n,p}.$$

- A tensor $\mathcal{D} \in \mathbb{R}^{m\times n\times p}$ is called f-diagonal [19] if $\mathcal{D}(i,j,k) = 0$ for all $i \neq j$.

Definition 8 ([14, 19, 21]). *The c-SVD of a tensor $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ is a tensor factorization of the form*

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

where $\mathcal{U} \in \mathbb{R}^{m\times m\times p}$ and $\mathcal{V} \in \mathbb{R}^{m\times m\times p}$ are orthogonal tensors, and $\mathcal{S} \in \mathbb{R}^{m\times n\times p}$ is an f-diagonal tensor with tube magnitudes $\mathcal{S}(i,i,:)$ that decrease in descending order, i.e.,

$$\|s_1\|_2 \geq \|s_2\|_2 \geq \cdots \geq \|s_k\|_2 \geq 0,$$

where $s_j = vec(\mathcal{S}(j,j,:)) \in \mathbb{R}^p$ and $k = \min\{m,n\}$.

Figure 4 shows a graphical representation of the c-SVD for a tensor in $\mathbb{R}^{m\times n\times p}$.

Using Definition 4 and equation (4), we obtain

$$\mathcal{A} = L^{-1}(L(\mathcal{U}) \triangle L(\mathcal{S}) \triangle L(\mathcal{V}^T)). \tag{5}$$

Therefore, it follows from (5) that the c-SVD of a tensor $\mathcal{A}$ can be obtained by computing the matrix SVD on each frontal slice of $\overline{\mathcal{A}} = L(\mathcal{A})$.

Example 5. *Consider the tensor $\mathcal{A} \in \mathbb{R}^{2\times 3\times 2}$ from Example 4. The c-SVD of $\mathcal{A}$ is expressed as $\mathcal{A} = \mathcal{U} *_c \mathcal{S} *_c \mathcal{V}^T$, where the frontal slices of $\mathcal{U} \in \mathbb{R}^{2\times 2\times 2}$, $\mathcal{S} \in \mathbb{R}^{2\times 3\times 2}$, and $\mathcal{V} \in \mathbb{R}^{3\times 3\times 2}$ are as follows*

$$\mathcal{U}^{(1)} = \begin{pmatrix} -0.1258 & 1.0070 \\ 1.0070 & 0.1258 \end{pmatrix}, \quad \mathcal{U}^{(2)} = \begin{pmatrix} 0.9774 & 0.1221 \\ 0.1221 & -0.9774 \end{pmatrix},$$

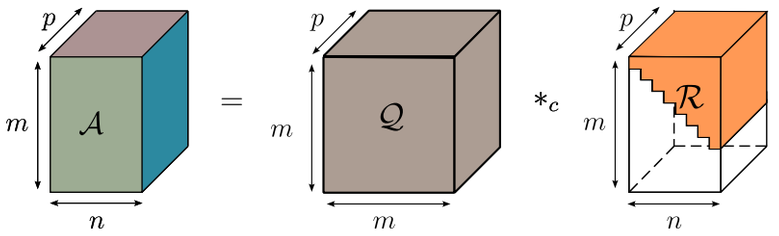


Figure 5: Graphical representation of the c-QR decomposition for $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$.

$$\mathcal{S}^{(1)} = \begin{pmatrix} 5.6847 & 0 & 0 \\ 0 & 0.7164 & 0 \end{pmatrix}, \quad \mathcal{S}^{(2)} = \begin{pmatrix} -1.7757 & 0 & 0 \\ 0 & 0.1321 & 0 \end{pmatrix},$$

and

$$\mathcal{V}^{(1)} = \begin{pmatrix} -0.6531 & -0.7237 & 0.8018 \\ 1.1248 & -0.3719 & -0.6999 \\ 0.2315 & -1.0491 & 0.6999 \end{pmatrix}, \quad \mathcal{V}^{(2)} = \begin{pmatrix} 0.1464 & 0.3968 & -0.4774 \\ 0.1775 & 0.0221 & 0.2735 \\ -0.4493 & -0.2816 & -0.2735 \end{pmatrix}.$$

Note that $\mathcal{U}$ and $\mathcal{V}$ are orthogonal tensors. Moreover, if we denote $s_1 = vec(\mathcal{S}(1,1,:)) = (5.6847,\ -1.7757)^T$ and $s_2 = vec(\mathcal{S}(2,2,:)) = (0.7164,\ 0.1321)^T$, then, $\|s_1\|_2 = 5.9556$ and $\|s_2\|_2 = 0.7285$. Therefore, it is evident that $\|s_1\|_2 \geq \|s_2\|_2 > 0$, according to Definition 8.

2.3.2. c-QR

The c-QR is a tensor QR decomposition based on the reduced c-product. Before introducing the c-QR, we first define an upper triangular tensor [22]. A tensor $\mathcal{R} \in \mathbb{R}^{m\times n\times p}$ is called upper triangular if $\mathcal{R}(i,j,k) = 0$ whenever $i > j$.

Definition 9 ([22]). *The c-QR decomposition of a tensor $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ is a factorization given by*

$$\mathcal{A} = \mathcal{Q} *_c \mathcal{R}, \tag{6}$$

where $\mathcal{Q} \in \mathbb{R}^{m\times m\times p}$ is an orthogonal tensor, and $\mathcal{R} \in \mathbb{R}^{m\times n\times p}$ is an upper triangular tensor.

Figure 5 provides a visual representation of the c-QR decomposition for a tensor in $\mathbb{R}^{m\times n\times p}$.

Similar to the process described in the c-SVD, Definition 4 and (6) allow us to express

$$\mathcal{A} = L^{-1}(L(\mathcal{Q}) \triangle L(\mathcal{R})). \tag{7}$$

From (7), we observe that the c-QR of a tensor $\mathcal{A}$ can be computed by applying the matrix QR decomposition to each frontal slice of $\overline{\mathcal{A}} = L(\mathcal{A})$.

Example 6. *Consider the tensor $\mathcal{A} \in \mathbb{R}^{2\times 3\times 2}$ from Example 4. The c-QR of $\mathcal{A}$ is expressed as $\mathcal{A} = \mathcal{Q} *_c \mathcal{R}$, where the frontal slices of $\mathcal{Q} \in \mathbb{R}^{2\times 2\times 2}$ and $\mathcal{R} \in \mathbb{R}^{2\times 3\times 2}$ are as follows*

$$\mathcal{Q}^{(1)} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathcal{Q}^{(2)} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix},$$

and

$$\mathcal{R}^{(1)} = \begin{pmatrix} -1.4142 & 0.7071 & 2.1213 \\ 0 & 0 & 0 \end{pmatrix}, \quad \mathcal{R}^{(2)} = \begin{pmatrix} 2.8284 & -4.2426 & -1.4142 \\ 0 & 0.7071 & 0.7071 \end{pmatrix}.$$

Note that $\mathcal{Q}$ is an orthogonal tensor since it satisfies the property $\mathcal{Q} *_c \mathcal{Q}^T = \mathcal{Q}^T *_c \mathcal{Q} = \mathcal{I}_{2,2}$. Furthermore, $\mathcal{R}$ is an upper triangular tensor.

2.3.3. Tensor square root

Definition 10 ([23]). *The tensor square root of $\mathcal{A} \in \mathbb{R}^{m\times m\times p}$ under the reduced c-product is defined as the tensor $\mathcal{B} \in \mathbb{R}^{m\times m\times p}$ that satisfies $\mathcal{A} = \mathcal{B} *_c \mathcal{B}$. This tensor is denoted by $\mathcal{B} = \mathcal{A}^{1/2}$.*

From Definitions 4 and 10, the tensor square root $\mathcal{B} = \mathcal{A}^{1/2}$ can be computed by taking the matrix square root of each frontal slice of $\overline{\mathcal{A}} = L(\mathcal{A})$. This follows from the relationship

$$\mathcal{A} = L^{-1}(L(\mathcal{B}) \triangle L(\mathcal{B})).$$

Furthermore, a tensor $\mathcal{A}$ possesses a tensor square root if and only if each frontal slice $\overline{\mathcal{A}}^{(i)}$ is symmetric and positive definite. For more details, refer to [23].

Example 7. *Consider the tensor $\mathcal{A} \in \mathbb{R}^{2\times 2\times 3}$, whose frontal slices are given by*

$$\mathcal{A}^{(1)} = \begin{pmatrix} 6.989 & 1.711 \\ 1.711 & 7.855 \end{pmatrix}, \quad \mathcal{A}^{(2)} = \begin{pmatrix} -4.222 & 0.676 \\ 0.676 & 3.802 \end{pmatrix}, \quad \mathcal{A}^{(3)} = \begin{pmatrix} 4.161 & 4.540 \\ 4.540 & 2.198 \end{pmatrix}.$$

Since the frontal slices $\overline{\mathcal{A}}^{(1)}$, $\overline{\mathcal{A}}^{(2)}$, and $\overline{\mathcal{A}}^{(3)}$ are symmetric and positive definite, the tensor $\mathcal{A}$ admits a tensor square root. Moreover, this tensor square root is given by $\mathcal{B} \in \mathbb{R}^{2\times 2\times 3}$, with frontal slices

$$\mathcal{B}^{(1)} = \begin{pmatrix} 3.0593 & 0.2955 \\ 0.2955 & 3.2304 \end{pmatrix}, \quad \mathcal{B}^{(2)} = \begin{pmatrix} -1.2316 & 0.0634 \\ 0.0634 & 0.8699 \end{pmatrix}, \quad \mathcal{B}^{(3)} = \begin{pmatrix} 1.2708 & 1.1901 \\ 1.1901 & 0.5471 \end{pmatrix}.$$

Finally, note that $\mathcal{A} = \mathcal{B} *_c \mathcal{B}$, which implies that $\mathcal{B} = \mathcal{A}^{1/2}$.

2.4. Tubal-rank and multi-rank

In this section, we define two tensorial scalar operators, understanding a tensorial scalar operator as one that results in a scalar, in this case, a real number.

Definition 11 ([24]). *Let $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ and consider $\mathcal{A} = \mathcal{U} *_c \mathcal{S} *_c \mathcal{V}^T$ as the c-SVD of $\mathcal{A}$. The tubal rank of $\mathcal{A}$, denoted as $rank_t(\mathcal{A})$, is defined as the number of non-zero tubes in the tensor f-diagonal $\mathcal{S}$, i.e.,*

$$rank_t(\mathcal{A}) = \#\{i, \|s_i\|_2 \neq 0\},$$

where $s_i = vec(\mathcal{S}(i,i,:))$.

Example 8. *Consider the tensor $\mathcal{A} \in \mathbb{R}^{2\times 3\times 3}$, defined as $\mathcal{A} = \mathcal{B} *_c \mathcal{C}$. The frontal slices of $\mathcal{B} \in \mathbb{R}^{2\times 1\times 3}$ and $\mathcal{C} \in \mathbb{R}^{1\times 3\times 3}$ are given by*

$$\mathcal{B}^{(1)} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad \mathcal{B}^{(2)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \mathcal{B}^{(3)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

and

$$\mathcal{C}^{(1)} = \begin{pmatrix} -2 & -1 & 2 \end{pmatrix}, \quad \mathcal{C}^{(2)} = \begin{pmatrix} -2 & 2 & 0 \end{pmatrix}, \quad \mathcal{C}^{(3)} = \begin{pmatrix} 2 & -2 & 0 \end{pmatrix},$$

respectively. When computing the c-SVD of $\mathcal{A}$, the frontal slices of the singular value tensor $\mathcal{S} \in \mathbb{R}^{2\times 3\times 3}$ are

$$\mathcal{S}^{(1)} = \begin{pmatrix} 5.0478 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \mathcal{S}^{(2)} = \begin{pmatrix} -1.6591 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \mathcal{S}^{(3)} = \begin{pmatrix} -1.8074 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Using the notation from Definition 8, we have $s_1 = (5.0478, -1.6591, -1.8074)^T$ and $s_2 = (0,0,0)^T$, where $\|s_1\|_2 = 5.6124$ and $\|s_2\|_2 = 0$. Consequently, we conclude that $\text{rank}_t(\mathcal{A}) = 1$, because only the norm of s_1 is non-zero.

Definition 12 ([21]). *Consider the tensor $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ and $\overline{\mathcal{A}} = L(\mathcal{A})$. We define the multi-rank of $\mathcal{A}$ as the vector $\mathbf{r} = (r_1, r_2, \ldots, r_p)^T \in \mathbb{R}^p$, where $r_i = \text{rank}(\overline{\mathcal{A}}^{(i)})$ for all $i = 1, 2, \ldots, p$.*

Note that the tubal rank and the multi-rank are related because $\text{rank}_t(\mathcal{A}) = \max_{i=1,\ldots,p} r_i$.

Example 9. *Consider the tensor $\mathcal{A} \in \mathbb{R}^{2\times 3\times 3}$ from Example 8 and let $\overline{\mathcal{A}} = L(\mathcal{A})$. Here, $\text{rank}(\overline{\mathcal{A}}^{(1)}) = \text{rank}(\overline{\mathcal{A}}^{(2)}) = \text{rank}(\overline{\mathcal{A}}^{(3)}) = 1$. Therefore the multi-rank of $\mathcal{A}$ is $\mathbf{r} = (1,1,1)^T$.*

2.5. Generalized tensor inverses

In this section, we define three types of generalized tensor inverses under the reduced c-product: the tensor inverse, the tensor pseudoinverse, and the tensor Drazin inverse.

2.5.1. Tensor inverse

Definition 13 ([14]). *A tensor $\mathcal{A} \in \mathbb{R}^{n\times n\times p}$ is said to be invertible under the reduced c-product if there exists $\mathcal{A}^{-1} \in \mathbb{R}^{n\times n\times p}$ such that*

$$\mathcal{A} *_c \mathcal{A}^{-1} = \mathcal{A}^{-1} *_c \mathcal{A} = \mathcal{I}_{n,p}. \tag{8}$$

Here, $\mathcal{A}^{-1}$ is called the tensor inverse under the reduced c-product, or simply the c-inverse.

Applying the operator L to (8), together with Definition 4, results in

$$L(\mathcal{A})^{(i)} L(\mathcal{A}^{-1})^{(i)} = L(\mathcal{A}^{-1})^{(i)} L(\mathcal{A})^{(i)} = I_p, \tag{9}$$

for all $i = 1, \ldots, p$. Therefore, $\mathcal{A}$ is invertible if and only if each $L(\mathcal{A})^{(i)}$ is invertible.

Example 10. *Consider the tensor $\mathcal{A} \in \mathbb{R}^{2\times 2\times 2}$, whose frontal slices are given by*

$$\mathcal{A}^{(1)} = \begin{pmatrix} 1 & -1 \\ 0 & -1 \end{pmatrix}, \quad \mathcal{A}^{(2)} = \begin{pmatrix} 0 & 2 \\ -1 & -2 \end{pmatrix}.$$

The c-inverse of $\mathcal{A}$, denoted as $\mathcal{B} \in \mathbb{R}^{2\times 2\times 2}$, has frontal slices given by

$$\mathcal{B}^{(1)} = \begin{pmatrix} 1.75 & 1.25 \\ -0.75 & -0.25 \end{pmatrix}, \quad \mathcal{B}^{(2)} = \begin{pmatrix} 1.25 & -0.25 \\ -0.25 & -0.75 \end{pmatrix}.$$

Note that $\mathcal{A} *_c \mathcal{B} = \mathcal{B} *_c \mathcal{C} = \mathcal{I}_{2,2}$, and therefore $\mathcal{B} = \mathcal{A}^{-1}$.

2.5.2. Tensor pseudoinverse

Definition 14 ([19, 25]). *The tensor pseudoinverse under the reduced c-product of $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$, or simply the c-pseudoinverse, is defined as a tensor $\mathcal{B} \in \mathbb{R}^{n\times m\times p}$ satisfying all of the following four criteria*

- $\mathcal{A} *_c \mathcal{B} *_c \mathcal{A} = \mathcal{A}$
- $\mathcal{B} *_c \mathcal{A} *_c \mathcal{B} = \mathcal{B}$
- $(\mathcal{A} *_c \mathcal{B})^T = \mathcal{A} *_c \mathcal{B}$
- $(\mathcal{B} *_c \mathcal{A})^T = \mathcal{B} *_c \mathcal{A}$

The c-pseudoinverse is denoted by $\mathcal{B} = \mathcal{A}^\dagger$.

If $\mathcal{A} = \mathcal{U} *_c \mathcal{S} *_c \mathcal{V}^T$ represents the c-SVD of $\mathcal{A}$, it follows from [19] that

$$\mathcal{A}^\dagger = \mathcal{V} *_c \mathcal{S}^\dagger *_c \mathcal{U}^T. \tag{10}$$

In addition, if operator L is applied to (10) and using Definition 4, we obtain

$$\mathcal{A}^\dagger = L^{-1}(L(\mathcal{V}) \triangle L(\mathcal{S}^\dagger) \triangle L(\mathcal{U}^T)). \tag{11}$$

As explained in [19], it follows from (11) that $\mathcal{A}^\dagger$ can be computed by determining the matrix pseudoinverse of $L(\mathcal{A})^{(i)}$ for all $i = 1, \ldots, p$.

Example 11. *Consider the tensor $\mathcal{A} \in \mathbb{R}^{2\times 3\times 2}$ from Example 4. According to Example 5, the c-SVD of $\mathcal{A}$ is computed and it is given by $\mathcal{A} = \mathcal{U} *_c \mathcal{S} *_c \mathcal{V}^T$. Moreover, the frontal slices of $\mathcal{S}^\dagger \in \mathbb{R}^{3\times 2\times 2}$ are*

$$(\mathcal{S}^\dagger)^{(1)} = \begin{pmatrix} 0.3899 & 0 \\ 0 & 2.8902 \\ 0 & 0 \end{pmatrix}, \quad (\mathcal{S}^\dagger)^{(2)} = \begin{pmatrix} 0.1218 & 0 \\ 0 & -0.5329 \\ 0 & 0 \end{pmatrix}.$$

Therefore, from (10), the c-pseudoinverse is given by $\mathcal{A}^\dagger = \mathcal{V} *_c \mathcal{S}^\dagger *_c \mathcal{U}^T$, where the frontal slices of $\mathcal{A}^\dagger \in \mathbb{R}^{3\times 2\times 2}$ are

$$(\mathcal{A}^\dagger)^{(1)} = \begin{pmatrix} -1.0622 & -1.0144 \\ -0.4593 & 0.0478 \\ -1.5407 & -0.0478 \end{pmatrix}, \quad (\mathcal{A}^\dagger)^{(2)} = \begin{pmatrix} 0.5167 & 1.1962 \\ 0.2775 & 0.6794 \\ -0.2775 & 1.3206 \end{pmatrix}.$$

2.5.3. Drazin c-inverse

Before introducing the Drazin tensor inverse, we first define the concept of a multi-index, which is essential for its definition.

Definition 15 ([26]). *Consider the tensors $\mathcal{A} \in \mathbb{R}^{m\times m\times p}$ and $\overline{\mathcal{A}} = L(\mathcal{A})$. We define the multi-index of $\mathcal{A}$ as the vector $\mathbf{t} = (t_1, t_2, \ldots, t_p)^T \in \mathbb{R}^p$, where t_j is the smallest non-negative integer satisfying*

$$rank((\overline{\mathcal{A}}^{(j)})^{t_j}) = rank((\overline{\mathcal{A}}^{(j)})^{t_j+1}),$$

for all $j = 1, 2, \ldots, p$.

Definition 16 ([26, 27]). *The Drazin tensor inverse, or simply the Drazin c-inverse, of a tensor $\mathcal{A} \in \mathbb{R}^{m\times m\times p}$ is the unique tensor $\mathcal{B} \in \mathbb{R}^{m\times m\times p}$ that satisfies the following conditions:*

- $\mathcal{B} *_c \mathcal{A} *_c \mathcal{B} = \mathcal{B}$
- $\mathcal{A} *_c \mathcal{B} = \mathcal{B} *_c \mathcal{A}$
- $\mathcal{B} *_c \mathcal{A}^{k+1} = \mathcal{A}^k$

where k is the tubal index of $\mathcal{A}$, defined as $k = \max\limits_{i=1,\ldots,p} t_i$, with $\mathbf{t} = (t_1, t_2, \ldots, t_p)^T$ being the multi-index of $\mathcal{A}$. The Drazin c-inverse is denoted by $\mathcal{B} = \mathcal{A}^D$.

As indicated in [26], the process of obtaining $\mathcal{A}^D$ involves computing the Drazin inverse of each frontal slice of $\overline{\mathcal{A}}$. For a more detailed description of the Drazin inverse for matrices, see [28], Chapter 4 of [29] and Result 6.6.7 in [30].

Example 12. *Consider the tensor $\mathcal{A} \in \mathbb{R}^{3\times3\times3}$, whose frontal slices are given by*

$$\mathcal{A}^{(1)} = \begin{pmatrix} 4 & -4 & -1 \\ -7 & -8 & 7 \\ -1 & -2 & 0 \end{pmatrix}, \quad \mathcal{A}^{(2)} = \begin{pmatrix} -2 & 2 & 1 \\ 4 & 4 & -4 \\ 0 & 1 & 0 \end{pmatrix}, \quad \mathcal{A}^{(3)} = \begin{pmatrix} -1 & 2 & 0 \\ 3 & 4 & -2 \\ 1 & 1 & 0 \end{pmatrix}.$$

The multi-index of $\mathcal{A}$ is the vector $\mathbf{t} = (2, 0, 0)^T$, and thus, the tubal index of $\mathcal{A}$ is $k = 2$. By computing the Drazin inverse of each frontal slice of $\overline{\mathcal{A}} = L(\mathcal{A})$, we obtain the Drazin c-inverse of $\mathcal{A}$, denoted as $\mathcal{A}^D \in \mathbb{R}^{3\times3\times3}$, with frontal slices given by

$$(\mathcal{A}^D)^{(1)} = \begin{pmatrix} 1.3503 & 0.0672 & -0.9694 \\ -0.0759 & -0.0084 & -0.4810 \\ 0.2435 & 0.2412 & -1.4518 \end{pmatrix}, (\mathcal{A}^D)^{(2)} = \begin{pmatrix} 0.5273 & -0.1091 & 1.3818 \\ 0 & 0 & 0.6667 \\ -0.4364 & -0.2545 & 1.8909 \end{pmatrix},$$

$$(\mathcal{A}^D)^{(3)} = \begin{pmatrix} 1.1224 & 0.0419 & -0.4124 \\ 0.0759 & 0.0084 & -0.1857 \\ 0.1929 & 0.0133 & -0.4391 \end{pmatrix}.$$

2.6. Tensor norms

In this section, we introduce three commonly used tensor norms based on the reduced c-product: the Frobenius norm, the spectral norm, and the nuclear norm.

Definition 17. *Let $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ be a tensor.*

- ***Frobenius norm** [13]: The Frobenius norm of $\mathcal{A}$ is defined as*

$$\|\mathcal{A}\|_{fr} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{p}(\mathcal{A}(i,j,k))^2}.$$

- ***Spectral norm** [26]: The spectral norm of $\mathcal{A}$ is defined as*

$$\|\mathcal{A}\|_2 = \max_{i=1,\dots,p} \|\overline{\mathcal{A}}^{(i)}\|_2.$$

- ***Nuclear norm** [18]: The nuclear norm of $\mathcal{A}$ is defined as*

$$\|\mathcal{A}\|_* = \sum_{i=1}^{p}\sum_{j=1}^{r}\overline{\sigma}_j^{(i)},$$

where $r = \min\{m,n\}$, and $\overline{\sigma}_1^{(i)} \geq \overline{\sigma}_2^{(i)} \geq \cdots \geq \overline{\sigma}_r^{(i)} \geq 0$ are the singular values of $\overline{\mathcal{A}}^{(i)}$.

Example 13. *Consider the tensor $\mathcal{A} \in \mathbb{R}^{2\times 3\times 2}$ from Example 4. Using the formulas given in Definition 17, we obtain $\|\mathcal{A}\|_{fr} = 6$, $\|\mathcal{A}\|_2 = 5.2753$ and $\|\mathcal{A}\|_* = 9.0525$.*

2.7. Tensor optimization problems

Finally, in this section, we explain three of the most commonly used tensor optimization problems based on the reduced c-product, particularly in problems related to signal and image processing.

2.7.1. Low-tubal-rank tensor approximation problem

The low-tubal-rank tensor approximation (LTRTA) problem for a tensor $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ involves finding a tensor $\mathcal{A}_r \in \mathbb{R}^{m\times n\times p}$ with $\text{rank}_t(\mathcal{A}) \leq r$ such that the following problem is minimized

$$\min_{\mathcal{A}_r} \|\mathcal{A} - \mathcal{A}_r\|_{fr}^2. \tag{12}$$

It follows from [24] that if $\mathcal{A} = \mathcal{U} *_c \mathcal{S} *_c \mathcal{V}^T$ is the c-SVD of $\mathcal{A}$, the solution of problem (12) is given by

$$\mathcal{A}_r = \mathcal{U}_r *_c \mathcal{S}_r *_c \mathcal{V}_r^T,$$

where $\mathcal{U}_r = \mathcal{U}(:,1:r,:) \in \mathbb{R}^{m\times r\times p}$, $\mathcal{S}_r = \mathcal{S}(1:r,1:r,:) \in \mathbb{R}^{r\times r\times p}$, and $\mathcal{V}_r = \mathcal{V}(:,1:r,:) \in \mathbb{R}^{r\times n\times p}$. The LTRTA problem has been applied to various real-world problems, including image completion and denoising [31], as well as the recovery of third-order tensors under random sampling [32].

Example 14. *Consider the tensor $\mathcal{A} \in \mathbb{R}^{2\times3\times2}$ from Example 4. Using $r = 1$ and the c-SVD of $\mathcal{A}$ computed in Example 5, the solution of the LTRTA problem is $\mathcal{A}_r \in \mathbb{R}^{2\times3\times2}$, whose frontal slices are given by*

$$\mathcal{A}_r^{(1)} = \begin{pmatrix} 1.2231 & -0.8653 & -1.5821 \\ -1.8785 & 3.0003 & 0.8833 \end{pmatrix}, \quad \mathcal{A}_r^{(2)} = \begin{pmatrix} -2.0665 & 3.0329 & 1.2195 \\ 0.7604 & -0.1267 & -1.3641 \end{pmatrix},$$

where $rank_t(\mathcal{A}_1) = 1$.

2.7.2. Tensor singular value thresholding problem

For $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$, the tensor singular value thresholding (TSVT) problem aims to find a tensor $\mathcal{X} \in \mathbb{R}^{m\times n\times p}$ that minimizes the following optimization problem

$$\min_{\mathcal{X}} \tau\|\mathcal{X}\|_* + \frac{1}{2}\|\mathcal{X} - \mathcal{A}\|_{fr}^2, \tag{13}$$

where $\tau > 0$ is a given constant. Let $\mathcal{A} = \mathcal{U} *_c \mathcal{S} *_c \mathcal{V}^T$ be the c-SVD decomposition of $\mathcal{A}$, then it follows from [33] that a solution of the TSVT problem is given by

$$\mathcal{X} = \mathcal{U} *_c \mathcal{S}_\tau *_c \mathcal{V}^T,$$

where $\mathcal{S}_\tau = L^{-1}\big((L(\mathcal{S}) - \tau)_+\big) \in \mathbb{R}^{m\times n\times p}$. Here, $t_+ = \max(t, 0)$. Problem (13) has been applied to image recovery and synthetic data problems [34, 33].

Example 15. *Consider the tensor $\mathcal{A} \in \mathbb{R}^{2\times3\times2}$ from Example 4. Using $\tau = 0.5$ and the c-SVD of $\mathcal{A}$ computed in Example 5, the solution of the TSVT problem is $\mathcal{X} \in \mathbb{R}^{2\times3\times2}$, whose frontal slices are given by*

$$\mathcal{X}^{(1)} = \begin{pmatrix} 1.1305 & -0.8906 & -1.4696 \\ -1.6425 & 2.6027 & 0.8603 \end{pmatrix}, \quad \mathcal{X}^{(2)} = \begin{pmatrix} -1.8474 & 2.6381 & 1.0664 \\ 0.7463 & -0.2279 & -1.1740 \end{pmatrix}.$$

2.7.3. Tensor least squares problem

Given tensors $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$, $\mathcal{B} \in \mathbb{R}^{m\times r\times p}$, and $\mathcal{C} \in \mathbb{R}^{s\times n\times p}$, the tensor least squares (TLS) problem consists of finding the tensor $\mathcal{X} \in \mathbb{R}^{r\times s\times p}$ that minimizes the following problem

$$\min_{\mathcal{X}} \|\mathcal{A} - \mathcal{B} *_c \mathcal{X} *_c \mathcal{C}\|_{fr}^2. \tag{14}$$

As demonstrated in [35], a solution of problem (14) is given by

$$\mathcal{X} = \mathcal{B}^\dagger *_c \mathcal{A} *_c \mathcal{C}^\dagger. \tag{15}$$

The TLS problem has been applied to noise reduction in audio and images [9]. Additionally, it has been used in linear regression with applications to the higher-order Gauss–Markov theorem [35].

Example 16. *Consider the tensors* $\mathcal{A} \in \mathbb{R}^{2\times3\times2}$, $\mathcal{B} \in \mathbb{R}^{2\times2\times2}$ *and* $\mathcal{C} \in \mathbb{R}^{1\times3\times2}$, *with their frontal slices defined as follow*

$$A^{(1)} = \begin{pmatrix} 1 & -1 & -1 \\ 0 & -2 & -3 \end{pmatrix}, \quad A^{(2)} = \begin{pmatrix} -2 & 2 & 1 \\ 1 & 0 & -1 \end{pmatrix},$$

$$B^{(1)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad B^{(2)} = \begin{pmatrix} 0 & 3 \\ 2 & 0 \end{pmatrix},$$

and

$$C^{(1)} = \begin{pmatrix} 0 & -2 & 1 \end{pmatrix}, \quad C^{(2)} = \begin{pmatrix} 1 & -1 & 1 \end{pmatrix}.$$

Using formula (15), *the solution of the TLS problem is the tensor* $\mathcal{X} \in \mathbb{R}^{2\times1\times2}$, *where its frontal slices*

$$X^{(1)} = \begin{pmatrix} -0.8857 \\ -0.4 \end{pmatrix}, \quad X^{(2)} = \begin{pmatrix} 0.9143 \\ 0.2 \end{pmatrix}.$$

3. Functions in the *C-Product Toolbox*

This section presents the functions included in the *C-Product Toolbox*, designed for use in both MATLAB and Python. All functions in the toolbox are based on the theory described in Section 2, and the package is available on GitHub at

https://github.com/jusotoTEC/c-product-toolbox.

Moreover, all the numerical examples presented in Section 2 were computationally implemented in both MATLAB and Python. These implementations are available in the *C-Product Toolbox* within a file named `test_toolbox`.

We present a detailed list of the functions implemented in the *C-Product Toolbox* in Table 2, including brief descriptions and their syntax. These functions are designed to simplify working with third-order tensors, making the toolbox particularly valuable for researchers and professionals in fields such as multidimensional data processing, tensor decomposition, and other applications related to the analysis of complex data structures. Moreover, the functionality of all the functions in this toolbox is analogous to that of similar functions available in MATLAB or Python.

3.1. Some considerations for commands `csvd`, `cqr`, `tubalrank`, `cdrazin` *and* `cnorm`

Some of the functions presented in Table 2 have specific considerations, which are explained below.

- The `csvd` command includes a second parameter, `opt`, which specifies the dimensions of the output tensors. If `opt='full'`, the result is the full c-SVD. In this case, the dimensions of the output tensors are $\mathcal{U} \in \mathbb{R}^{m\times m\times p}$, $\mathcal{S} \in \mathbb{R}^{m\times n\times p}$, and $\mathcal{V} \in \mathbb{R}^{n\times n\times p}$. If `opt='econ'`, the result is the economy-sized c-SVD. For this option, assuming $s = \min\{m, n\}$, the dimensions of the output tensors are $\mathcal{U} \in \mathbb{R}^{m\times s\times p}$, $\mathcal{S} \in \mathbb{R}^{s\times s\times p}$, and $\mathcal{V} \in \mathbb{R}^{s\times n\times p}$. If the `opt` parameter is not specified, the default value is `opt='full'`.

Function	Description	Syntax	Reference
`cprod`	Reduced c-product	`C=cprod(A,B)`	Definition 4
`cinprod`	Tensor inner product	`c=cinprod(A,B)`	Definition 5
`ceye`	Identity tensor	`I=ceye(n,p)`	Definition 6
`ctransp`	Tensor transpose	`B = ctransp(A)`	Definition 7
`csvd`	c-SVD	`[U,S,V]=csvd(A,opt)`	Definition 8
`cqr`	c-QR	`[Q,R]=cqr(A,opt)`	Definition 9
`csqrtt`	Tensor square root	`X=csqrtt(A)`	Definition 10
`ctubalrank`	Tubal-rank	`trank=ctubalrank(A)`	Definition 11
`cmultirank`	Multi-rank	`mrank=cmultirank(A)`	Definition 12
`cinv`	c-inverse	`X=cinv(A)`	Definition 13
`cpinv`	c-pseudoinverse	`X=cpinv(A)`	Definition 14
`cdrazin`	Drazin c-inverse and multi-index	`[X,t]=cdrazin(A)`	Definitions 15 and 16
`cnorm`	Tensor norm	`z=cnorm(A,opt)`	Definition 17
`clowrank`	LTRTA problem	`Ar=clowrank(A,r)`	Section 2.7.1
`csvt`	TSVT problem	`D=csvt(A,t)`	Section 2.7.2
`clsq`	TLS problem	`X=tlsq(A,B,C)`	Section 2.7.3
`test_toolbox`	Numerical examples	`test_toolbox()`	Examples in Section 2

Table 2: A list of function in the *C-Product Toolbox*.

- The `cqr` command also includes a second parameter, `opt`, which specifies the dimensions of the output tensors. If `opt='full'`, the result is the full c-QR decomposition. In this case, the dimensions of the output tensors are $\mathcal{Q} \in \mathbb{R}^{m\times m\times p}$ and $\mathcal{R} \in \mathbb{R}^{m\times n\times p}$. If `opt='econ'`, the result is the economy-sized c-QR decomposition. For $m > n$, the dimensions of the output tensors are $\mathcal{Q} \in \mathbb{R}^{m\times n\times p}$ and $\mathcal{R} \in \mathbb{R}^{n\times n\times p}$. If $m \leq n$, the economy-sized decomposition is identical to the full decomposition. If the `opt` parameter is not specified, the default value is `opt='full'`.

- The `tubalrank` command computes the tubal-rank as the number of tubes, $\mathcal{S}(i,i,:)$, whose Euclidean norms exceed a specified tolerance. This concept is formally defined as:

$$\texttt{tubalrank}(\mathcal{A}) = \#\{i, \|s_i\|_2 > \max\{m,n\} \cdot \texttt{eps} \cdot \|s_1\|_2\}, \tag{16}$$

  where $s_i = \text{vec}(\mathcal{S}(i,i,:))$ and $\texttt{eps} = 2^{-52}$ represents machine precision in a double-precision floating-point system[1]. The implementation presented in (16) is analogous to numerical methods for estimating matrix rank, such as the `rank` function in MATLAB or Python using the Numpy or Scipy libraries.

[1]Machine precision refers to the smallest distinguishable difference between two floating-point numbers in the system.

- The `cdrazin` command computes the Drazin inverse for matrices (or simply Drazin inverse) of each frontal slice of $\overline{\mathcal{A}}$ to obtain the Drazin c-inverse, as described in [26]. Various methods exist for computing the Drazin inverse of a matrix. In the *C-Product Toolbox*, the `cdrazin` command implements the formula provided in [28, Theorem 5] for computing the Drazin inverse. Specifically, for a matrix $A \in \mathbb{R}^{m\times m}$ with tubal index k, the Drazin inverse $A^D \in \mathbb{R}^{m\times m}$ is computed as $A^D = A^k(A^{2k+1})^\dagger A^k$. This formula is used in the toolbox due to its simplicity and computational efficiency.

- The `cnorm` command also accepts a second parameter, `opt`, which specifies the type of tensor norm to compute. When `opt='fro'`, the `cnorm` function calculates the Frobenius norm. For `opt='spec'`, it computes the spectral norm. Similarly, when `opt='nuc'`, the function calculates the nuclear norm. If the `opt` parameter is not specified, the default value is `opt='fro'`.

4. Numerical Experiments

This section presents several numerical experiments that demonstrate the advantages and efficiency of the proposed *C-Product Toolbox*. The experiments were conducted on a desktop computer with an Intel(R) Core(TM) i9-10900F processor (2.80 GHz) and 32 GB of RAM, running MATLAB R2021a. The corresponding MATLAB codes are available in the GitHub repository of the *C-Product Toolbox* at https://github.com/jusotoTEC/c-product-toolbox.

4.1. Numerical experiment 1: Benchmarking memory usage of the t-product, c-product, and reduced c-product

In this experiment, we compare the memory usage of three tensor products: the t-product [13], the c-product [14], and the proposed reduced c-product in Algorithm 1. To do this, we generate two tensors, $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ and $\mathcal{B} \in \mathbb{R}^{n\times m\times p}$, where $n = m/2$ and $p = m/10$, considering values of $m = 100k$ for $k = 5, \ldots, 15$. We compute the tensor product between $\mathcal{A}$ and $\mathcal{B}$ using the three mentioned tensor products, and then compare the memory required for each computation. To measure memory usage, we calculate the total number of bytes used by all matrices involved in each tensor product. The memory usage is measured using the `whos` command in MATLAB, which returns information about each matrix as a nested structure array containing a scalar `struct` for each variable. Each scalar `struct` includes the field `bytes`, which stores the number of bytes allocated for the matrix. More details about the `whos` command can be found in [36].

Figure 6 shows the simulation results, displaying the tensor dimension m versus the memory used (in bytes) for computing the t-product, c-product, and reduced c-product. It is evident from Figure 6 that the proposed reduced c-product requires less memory compared to both the t-product and the c-product. Specifically, the memory used by the reduced c-product corresponds to only 67% of the memory required by the t-product and 50% of that required by the c-product, approximately. This reduction in memory usage is primarily due to two reasons:

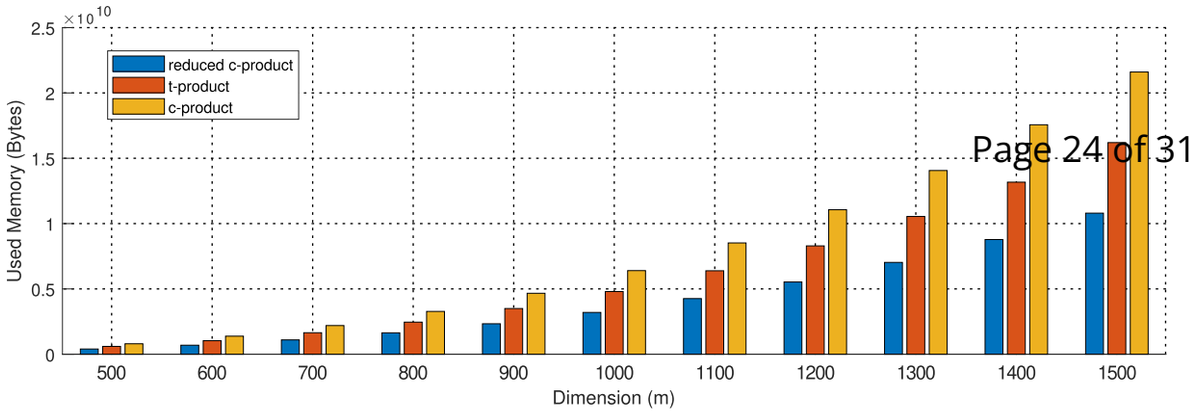


Figure 6: Bar chart of tensor dimension (m) versus memory usage (bytes) of the t-product, c-product, and reduced c-product.

- First, in the case of the t-product, computing $\overline{\mathcal{A}} = L(\mathcal{A})$ involves applying the DFT, which results in a tensor $\overline{\mathcal{A}}$ with complex-valued entries. Since complex numbers require twice the storage of real numbers (storing both real and imaginary components), this increases memory consumption using the t-product.

- Second, the c-product involves additional matrix computations, as shown in Table 1. In particular, the computation of $L(\mathcal{A})$ in the c-product requires extra matrices such as W, Z, and M, which are not needed for the reduced c-product. These additional matrices increase memory usage, whereas the reduced c-product avoids them, leading to more efficient memory consumption.

Based on the previous analysis and the numerical results shown in Figure 6, we conclude that the proposed reduced c-product requires less memory than the t-product and c-product. In some cases, it reduces memory usage by up to 50%.

4.2. Numerical experiment 2: Execution time comparison of the proposed C-Product Toolbox and the Tensor-Tensor Product Toolbox 2.0

In this section, we present the advantages of the proposed *C-Product Toolbox* in terms of execution time, speedup[2], and percent difference[3], comparing its performance against the corresponding functions from another toolbox in the literature, the *Tensor-Tensor Product Toolbox 2.0* [16]. This toolbox was designed for MATLAB and is based on the t-product as a tensor multiplication method. The toolbox presented in [16] is the only one in the

[2]Consider Method 1 and Method 2 that solve the same numerical problem, with execution times T_1 and T_2, respectively. As mentioned in [37], the speedup S (or acceleration) is defined as the ratio of their execution times $S = T_2/T_1$. If Method 1 outperforms Method 2, then S will be greater than 1. Conversely, if Method 1 degrades performance, S will be less than 1.

[3]As mentioned in [37], the percent difference P between Method 1 and Method 2, assuming $T_1 < T_2$, is given by $P = 100(T_2 - T_1)/T_2 = 100(1 - 1/S)$. This means that Method 1 is P% faster than Method 2.

Test case	Functions		Description
	C-Product Toolbox	***Tensor-Tensor Product Toolbox 2.0***	
1	`cprod`	`tprod`	Tensor product
2	`csvd`	`tsvd`	Tensor SVD
3	`cqr`	`tqr`	Tensor QR
4	`cinv`	`tinv`	Tensor inverse
5	`cnorm(·,'nuc')`	`tnn`	Tensor nuclear norm
6	`cnorm(·,'spec')`	`tsn`	Tensor spectral norm
7	`ctubalrank`	`tubalrank`	Tubal rank
8	`csvt`	`prox_tnn`	TSVT problem

Table 3: Description of the functions to be compared between the *C-Product Toolbox* and the *Tensor-Tensor Product Toolbox 2.0*.

literature that supports tensor operations where multiplication is based on invertible linear transformations, as defined in Definition 3, similar to the proposed *C-Product Toolbox*. For this reason, we focus our execution time comparison on these two toolboxes.

Although there are other MATLAB and Python toolboxes for tensor computations, such as *Tensor Toolbox for MATLAB* [38], *TensorFlow* [39], and *TensorLy* [40], their tensor multiplication methods do not rely on invertible linear transformations, as defined in Definition 3. In contrast, both the *C-Product Toolbox* and the *Tensor-Tensor Product Toolbox 2.0* are designed under this framework. Therefore, our comparison is limited to the *Tensor-Tensor Product Toolbox 2.0*, and all experiments are conducted in MATLAB, as this toolbox does not have a Python implementation.

In this numerical experiment, we compare 8 functions from each of the two toolboxes mentioned previously. These functions are listed in Table 3. Additionally, Table 4 provides the details of the tensors, dimensions, and corresponding parameters for the development of each test case.

Table 5 presents the diagrams of the numerical simulations of each test case under consideration. The results in Table 5 demonstrate the computational advantage of the *C-Product Toolbox* over the *Tensor-Tensor Product Toolbox 2.0* for most tested functions. Specifically, `cprod`, `csvd`, `cqr`, `cinv`, and `csvt` exhibit consistently lower execution times, achieving speedups greater than 1 in all simulations, with performance improvements ranging from 10% to 70%. For functions `cnorm(·,'nuc')`, `cnorm(·,'spec')` and `ctubalrank`, execution times are comparable between the two toolboxes. However, in most cases, the *C-Product Toolbox* still achieves better performance in terms of percent difference, with only a few exceptions where the *Tensor-Tensor Product Toolbox 2.0* is slightly more efficient. Overall, these results highlight the efficiency and potential benefits of the *C-Product Toolbox* in tensor computations.

Test case	Tensors	Dimensions	Parameter m	Ranges for k
1	$\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ $\mathcal{B} \in \mathbb{R}^{n\times s\times p}$	$n = \lfloor \frac{m}{4} \rfloor$, $s = \lfloor \frac{m}{3} \rfloor$, $p = \lfloor \frac{m}{5} \rfloor$	$m = 125k + 500$	$k = 0, 1, \ldots, 12$
2	$\mathcal{A} \in \mathbb{R}^{m\times n\times p}$	$n = \lfloor \frac{m}{4} \rfloor$, $p = \lfloor \frac{m}{8} \rfloor$	$m = 125k + 500$	$k = 0, 1, \ldots, 10$
3	$\mathcal{A} \in \mathbb{R}^{m\times n\times p}$	$n = \lfloor \frac{m}{5} \rfloor$, $p = \frac{m}{10} \rfloor$	$m = 125k + 500$	$k = 0, 1, \ldots, 11$
4	$\mathcal{A} \in \mathbb{R}^{m\times m\times p}$	$p = \lfloor \frac{m}{20} \rfloor$	$m = 125k + 500$	$k = 0, 1, \ldots, 12$
5 and 6	$\mathcal{A} \in \mathbb{R}^{m\times n\times p}$	$n = \lfloor \frac{m}{50} \rfloor$, $p = \lfloor \frac{m}{5} \rfloor$	$m = 125k + 500$	$k = 0, 1, \ldots, 28$
7	$\mathcal{A} \in \mathbb{R}^{m\times n\times p}$	$n = \lfloor \frac{m}{50} \rfloor$, $p = \lfloor \frac{m}{20} \rfloor$	$m = 125k + 500$	$k = 0, 1, \ldots, 36$
8	$\mathcal{A} \in \mathbb{R}^{m\times n\times p}$	$n = \lfloor \frac{m}{30} \rfloor$, $p = \lfloor \frac{m}{5} \rfloor$	$m = 125k + 500$	$k = 0, 1, \ldots, 20$

Table 4: Summary of tensors, dimensions, and corresponding parameters for the development of each Test Case in Table 3. Here, $\lfloor x \rfloor$ denotes the floor function of a number x, which represents the greatest integer less than or equal to x.

4.3. Numerical experiment 3: Applying the C-Product Toolbox for video denoising

This section presents an application of the *C-Product Toolbox* for video denoising. We consider the grayscale video `'original_video.avi'`, obtained from [41]. The video consists of 3488 frames, each with a resolution of 135×240. Table 6 displays four sample frames of this video. Notably, `'original_video.avi'` can be represented as a third-order tensor $\mathcal{A} \in \mathbb{R}^{135\times 240\times 3488}$, where the i-th frontal slice $A(:,:,i)$ corresponds to the i-th frame of the video.

We generate a noisy version of $\mathcal{A}$, denoted by $\mathcal{C} \in \mathbb{R}^{135\times 240\times 3488}$, as $\mathcal{C} = \mathcal{A} + 0.1\mathcal{N}$, where $\mathcal{N} \in \mathbb{R}^{135\times 240\times 3488}$ is a third-order tensor with entries drawn from a normal distribution with zero mean and standard deviation 1. Table 6 displays four sample frames of the noisy video.

Following [9, 42], we aim to remove noise from $\mathcal{C}$ by computing a tensor filter $\mathcal{X} \in \mathbb{R}^{135\times 135\times 3488}$ that solves the optimization problem

$$\min_{\mathcal{X}} \|\mathcal{A} - \mathcal{X} *_c \mathcal{C}\|_{fr}^2. \tag{17}$$

The tensor optimization problem presented in (17) has been previously studied in its matrix form, with applications in signal and image processing, as discussed, for example, in [43, 44].

Note that problem (17) is equivalent to the TLS problem in (14) when $\mathcal{B} = \mathcal{I}_{135,3488}$. Therefore, the solution to problem (17) is given by $\mathcal{X} = \mathcal{A} *_c \mathcal{C}^\dagger$, and thus, we can use the `clsq` command from the *C-Product Toolbox* to obtain the tensor filter $\mathcal{X}$. Table 6 presents estimates of four selected frames from the restored video $\widehat{\mathcal{A}} = \mathcal{X} *_c \mathcal{C}$. In addition, we also present in Table 6 the result of filtering the noisy image using the same procedure described above, but based on the t-product instead, as explained in [9].

To evaluate the quality of the reconstructed frames, we use the structural similarity index[4] (SSIM). Figure 7 displays the SSIM values comparing the original video (denoted by

[4]The structural similarity (SSIM) index is a metric for quantifying the perceived quality of digital images and videos. The SSIM index ranges from -1 to 1, where 1 is achieved only when two datasets are identical. More details can be found in [45].

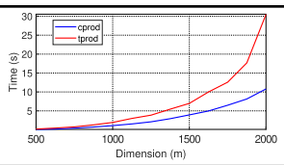
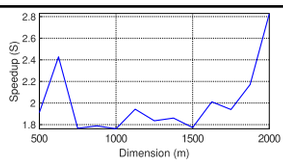
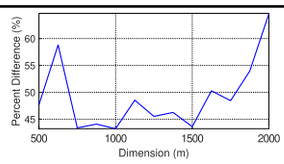
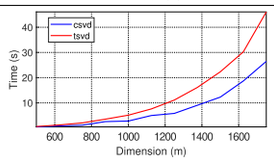
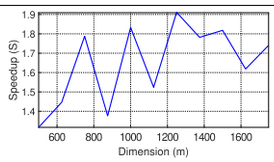
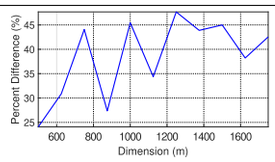
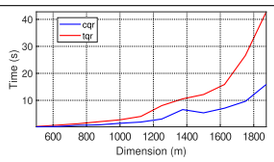
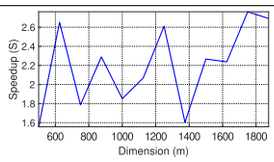
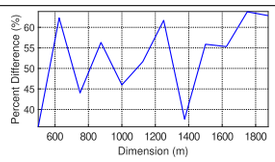
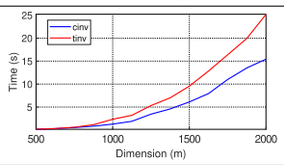
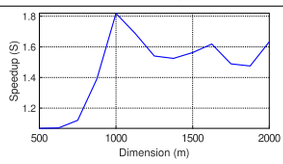
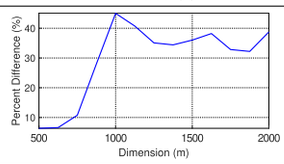
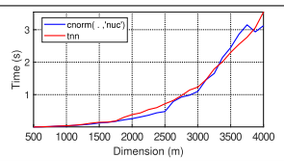
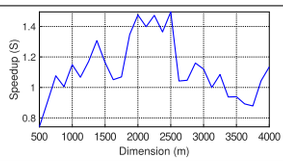
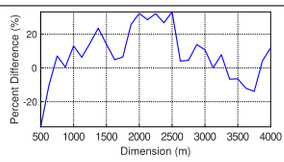
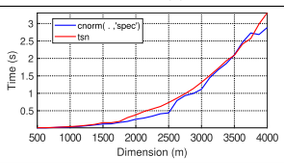
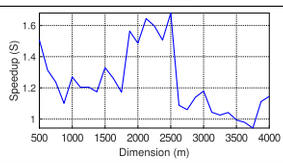
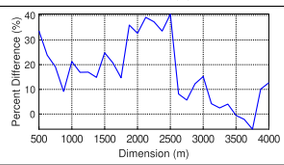
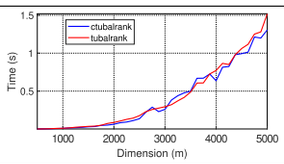
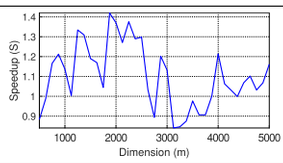
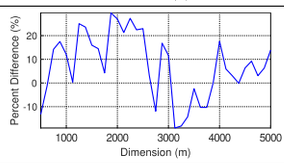
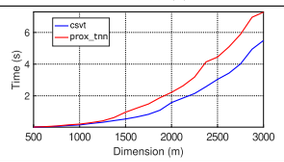
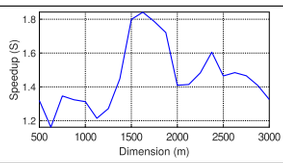
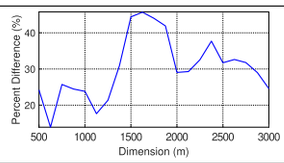
Test case	Time	Speedup	Percent difference
1			
2			
3			
4			
5			
6			
7			
8			

Table 5: Diagrams comparing execution time, speedup, and percent difference for eight functions from the *C-Product Toolbox* against their counterparts in the *Tensor-Tensor Product Toolbox 2.0*.

$\mathcal{A}$) with both the noisy input (denoted by $\mathcal{C}$) and the denoised results obtained using the reduced c-product and the t-product (denoted by $\widehat{\mathcal{A}}$). As shown in Figure 7, the SSIM values between the original and the denoised videos (represented by the green and blue lines for the

# Frame	Original video ($\mathcal{A}$)	Noisy video ($\mathcal{C}$)	Denoised video ($\widehat{\mathcal{A}}$) (reduced c-product)	Denoised video ($\widehat{\mathcal{A}}$) (t-product)
112				
598				
2287				
2463				

Table 6: Representative frames from the original video ($\mathcal{A}$), the noisy video ($\mathcal{C}$), and the denoised video ($\widehat{\mathcal{A}}$), obtained using the reduced c-product and the t-product.

reduced c-product and the t-product, respectively) remain consistently close to 1 across all frames. This indicates a high degree of structural similarity and confirms the effectiveness of the reconstruction. In contrast, the SSIM values between the original and the noisy video (red line) range between 0.6 and 0.8, clearly showing the degradation introduced by noise. These results highlight the effectiveness of the *C-Product Toolbox* for video denoising. While both approaches (the reduced c-product and the t-product) achieve comparable quality, the reduced c-product offers significant advantages in terms of lower memory usage and faster execution time, as demonstrated in previous Numerical Experiments 1 and 2.

5. Conclusion

This paper introduced the *C-Product Toolbox*, a new computational package for performing tensor operations based on the reduced c-product. Unlike existing toolboxes, such as the *Tensor-Tensor Product Toolbox 2.0*, the proposed package offers improved computational efficiency by utilizing the DCT and real arithmetic, reducing memory usage and execution time. Through a series of numerical experiments, we demonstrated the advantages of the *C-Product Toolbox* over existing alternatives, highlighting its superior performance in terms of memory efficiency and computational speed. Furthermore, we provided an application in video processing, showcasing the toolbox's capability in practical scenarios such as video denoising.

Additionally, to provide a solid theoretical foundation, this work includes a review of key mathematical concepts related to tensors and the reduced c-product, ensuring completeness and clarity.

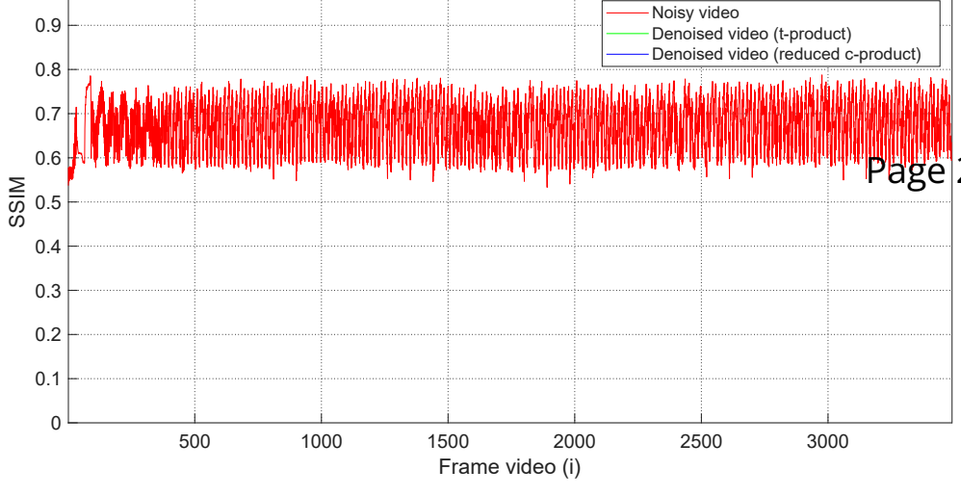


Figure 7: Diagram of the SSIM values comparing the original video to both the noisy and the denoised video.

Overall, this work contributes to the advancement of tensor-based computations by offering a specialized and optimized toolbox for third-order tensors. The *C-Product Toolbox* not only fills a gap in the availability of efficient tools for the reduced c-product but also serves as a foundation for further research and applications in fields such as signal and image processing.

Acknowledgements

This work was financially supported by *Vicerrectoría de Investigación y Extensión* from *Instituto Tecnológico de Costa Rica* (Research #1440054).

References

[1] C. Liu, H. Zhang, H. Fan, Y. Li, Tensorial bipartite graph clustering based on logarithmic coupled penalty, Pattern Recognition 156 (2024) 110860.

[2] J. Wang, T. Deng, M. Yang, Nonconvex submodule clustering via joint sliced sparse gradient and cluster-aware approach, Pattern Recognition (2024) 110619.

[3] P. Song, Z. Liu, J. Mu, Y. Cheng, Deep embedding based tensor incomplete multi-view clustering, Digital Signal Processing 151 (2024) 104534.

[4] C. Ozdemir, R. C. Hoover, K. Caudle, K. Braman, Tensor discriminant analysis on grassmann manifold with application to video based human action recognition, International Journal of Machine Learning and Cybernetics (2024) 1–13.

[5] X. Deng, Y. Shi, D. Yao, Theories, algorithms and applications in tensor learning, Applied Intelligence 53 (17) (2023) 20514–20534.

[6] F. Han, Y. Miao, Z. Sun, Y. Wei, T-ADAF: Adaptive data augmentation framework for image classification network based on tensor t-product operator, Neural Processing Letters 55 (8) (2023) 10993–11016.

[7] L. Reichel, U. O. Ugwu, Tensor Krylov subspace methods with an invertible linear transform product applied to image processing, Applied Numerical Mathematics 166 (2021) 186–207.

[8] M. El Guide, A. El Ichi, K. Jbilou, R. Sadaka, On tensor GMRES and Golub-Kahan methods via the t-product for color image processing, Electron. J. Linear Algebra 37 (2021) 524–543.
[9] P. Soto-Quiros, A least-squares problem of a linear tensor equation of third-order for audio and color image processing, in: 2022 45th International Conference on Telecommunications and Signal Processing (TSP), IEEE, 2022, pp. 59–65.
[10] L. Reichel, U. O. Ugwu, Weighted tensor Golub–Kahan–Tikhonov-type methods applied to image processing using a t-product, Journal of Computational and Applied Mathematics 415 (2022) 114488.
[11] C. D. Martin, R. Shafer, B. LaRue, An order-p tensor factorization with applications in imaging, SIAM Journal on Scientific Computing 35 (1) (2013) A474–A490.
[12] L. Sun, B. Zheng, C. Bu, Y. Wei, Moore–Penrose inverse of tensors via einstein product, Linear and Multilinear Algebra 64 (4) (2016) 686–698.
[13] M. E. Kilmer, C. D. Martin, Factorization strategies for third-order tensors, Linear Algebra and its Applications 435 (3) (2011) 641–658.
[14] E. Kernfeld, M. Kilmer, S. Aeron, Tensor–tensor products with invertible linear transforms, Linear Algebra and its Applications 485 (2015) 545–570.
[15] W. Qin, H. Wang, F. Zhang, J. Wang, X. Luo, T. Huang, Low-rank high-order tensor completion with applications in visual data, IEEE Transactions on Image Processing 31 (2022) 2433–2448.
[16] C. Lu, Tensor-Tensor Product Toolbox, Carnegie Mellon University, `https://github.com/canyilu/tproduct` (June 2018).
[17] C. Lu, Tensor-tensor product toolbox, arXiv preprint arXiv:1806.07247.
[18] C. Lu, X. Peng, Y. Wei, Low-rank tensor completion with a new tensor nuclear norm induced by invertible linear transforms, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 5996–6004.
[19] H. Jin, S. Xu, Y. Wang, X. Liu, The Moore–Penrose inverse of tensors via the m-product, Computational and Applied Mathematics 42 (6) (2023) 294.
[20] J. Chen, W. Huang, An iterative algorithm for low-rank tensor completion problem with sparse noise and missing values, Numerical Linear Algebra with Applications 31 (3) (2024) e2544.
[21] M. E. Kilmer, L. Horesh, H. Avron, E. Newman, Tensor-tensor algebra for optimal representation and compression of multiway data, Proceedings of the National Academy of Sciences 118 (28) (2021) e2015851118.
[22] Y. Zheng, A.-B. Xu, Tensor completion via tensor QR decomposition and l_2, 1-norm minimization, Signal Processing 189 (2021) 108240.
[23] L. Tang, Y. Yu, Y. Zhang, H. Li, Sketch-and-project methods for tensor linear systems, Numerical Linear Algebra with Applications 30 (2) (2023) e2470.
[24] M. Hached, K. Jbilou, C. Koukouvinos, M. Mitrouli, A multidimensional principal component analysis via the c-product Golub–Kahan–SVD for classification and face recognition, Mathematics 9 (11) (2021) 1249.
[25] H. Jin, M. He, Y. Wang, The expressions of the generalized inverses of the block tensor via the c-product, Filomat 37 (26) (2023) 8909–8926.
[26] J. K. Sahoo, S. K. Panda, R. Behera, P. S. Stanimirović, Computation of tensors generalized inverses under m-product and applications, Journal of Mathematical Analysis and Applications 542 (1) (2025) 128864.
[27] Y. Miao, L. Qi, Y. Wei, T-Jordan canonical form and t-Drazin inverse based on the t-product, Communications on Applied Mathematics and Computation 3 (2021) 201–220.
[28] R. E. Cline, Inverses of rank invariant powers of a matrix, SIAM Journal on Numerical Analysis 5 (1) (1968) 182–197.
[29] A. Ben-Israel, T. N. Greville, Generalized inverses: theory and applications, Springer Science & Business Media, 2006.
[30] D. S. Bernstein, Matrix mathematics: theory, facts, and formulas, Princeton university press, 2009.
[31] Z. Zhang, G. Ely, S. Aeron, N. Hao, M. Kilmer, Novel methods for multilinear data completion and denoising based on tensor-SVD, in: Proceedings of the IEEE conference on computer vision and pattern

recognition, 2014, pp. 3842–3849.
[32] Z. Zhang, S. Aeron, Exact tensor completion using t-SVD, IEEE Transactions on Signal Processing 65 (6) (2016) 1511–1526.
[33] M. Che, X. Wang, Y. Wei, X. Zhao, Fast randomized tensor singular value thresholding for low-rank tensor optimization, Numerical Linear Algebra with Applications 29 (6) (2022) e2444.
[34] C. Lu, J. Tang, S. Yan, Z. Lin, Nonconvex nonsmooth low rank minimization via iteratively reweighted nuclear norm, IEEE Transactions on Image Processing 25 (2) (2015) 829–839.
[35] H. Jin, M. Bai, J. Benítez, X. Liu, The generalized inverses of tensors and an application to linear models, Computers & Mathematics with Applications 74 (3) (2017) 385–397.
[36] MathWorks, Matlab reference documentation of command `whos`, 2025, 30 January.
URL `https://mathworks.com/help/matlab/ref/whos.html`
[37] H. Cragon, Computer Architecture and Implementation, Cambridge University Press, 2000.
[38] B. Bader, T. Kolda, et. al., Tensor Toolbox for MATLAB, Version 3.6, Sandia National Labs, `https://www.tensortoolbox.org/index.html` (September 2023).
[39] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., {TensorFlow}: a system for {Large-Scale} machine learning, in: 12th USENIX symposium on operating systems design and implementation (OSDI 16), 2016, pp. 265–283.
[40] J. Kossaifi, Y. Panagakis, A. Anandkumar, M. Pantic, Tensorly: Tensor learning in Python, Journal of Machine Learning Research 20 (26) (2019) 1–6.
[41] G. Korb, Video of cityscape during daytime, accessed on February 7, 2025 (2024).
URL `https://www.pexels.com/video/video-of-cityscape-during-daytime-4772894/`
[42] P. Soto-Quiros, Convergence analysis of iterative methods for computing the t-pseudoinverse of complete full-rank third-order tensors based on the t-product, Results in Applied Mathematics 18 (2023) 100372.
[43] M. A. Suliman, A. M. Alrashdi, T. Ballal, T. Y. Al-Naffouri, Snr estimation in linear systems with gaussian matrices, IEEE Signal Processing Letters 24 (12) (2017) 1867–1871.
[44] J. Chung, M. Chung, Computing optimal low-rank matrix approximations for image processing, in: 2013 Asilomar Conference on Signals, Systems and Computers, IEEE, 2013, pp. 670–674.
[45] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE transactions on image processing 13 (4) (2004) 600–612.