Universidade Federal do Ceará
Centro de Tecnologia
Departamento de Engenharia de Teleinformática
Engenharia de Teleinformática

Multilinear Algebra
Computational Homeworks

Student Kenneth Brenner dos Anjos Benício – 519189

Professor Andre Lima Ferrer de Almeida
Course Multilinear Algebra - TIP8419

Fortaleza, 2022

Homework 0
Kronecker Product Run Time

Run Time Perfomance of Sequential Kronecker Products

In here I will briefly analyze the run time performance of the inverse operator while also using the Kronecker Product. In the first case the number of products is fixed while the number of columns is varying. In the second case we have a varying number of products for a fixed number of columns. In both cases is possible to see that is preferable to first invert the matrices before applying the Kronecker operator.

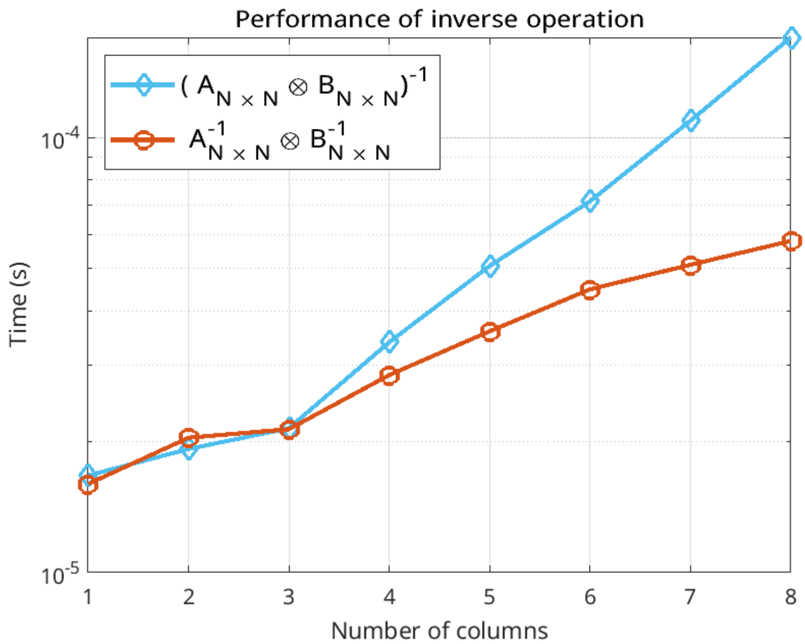

Figure 1: Monter Carlo Experiment with 5000.

Show that $\mathbf{eig}(\boldsymbol{A} \otimes \boldsymbol{B}) = \mathbf{eig}(\boldsymbol{A}) \otimes \mathbf{eig}(\boldsymbol{B})$

By using the eigenvalue decomposition (eig) of two matrices and apply the Kronecker Product to them it is possible to reach the intended result

$$\boldsymbol{A} \otimes \boldsymbol{B} = (\boldsymbol{C}_1 \boldsymbol{\Lambda_1} \boldsymbol{C}_1^{-1}) \otimes (\boldsymbol{C}_2 \boldsymbol{\Lambda_2} \boldsymbol{C}_2^{-1}), \tag{1}$$

$$\boldsymbol{A} \otimes \boldsymbol{B} = (\boldsymbol{C}_1 \boldsymbol{\Lambda_1} \otimes \boldsymbol{C}_2 \boldsymbol{\Lambda_2})(\boldsymbol{C}_2^{-1} \otimes \boldsymbol{C}_2^{-1}), \tag{2}$$

$$\boldsymbol{A} \otimes \boldsymbol{B} = (\boldsymbol{C}_1 \otimes \boldsymbol{C}_2)(\boldsymbol{\Lambda_1} \otimes \boldsymbol{\Lambda_2})(\boldsymbol{C}_2^{-1} \otimes \boldsymbol{C}_2^{-1}), \tag{3}$$

$$\mathrm{eig}(\boldsymbol{A} \otimes \boldsymbol{B}) = (\boldsymbol{\Lambda_1} \otimes \boldsymbol{\Lambda_2}) = \mathrm{eig}(\boldsymbol{A}) \otimes \mathrm{eig}(\boldsymbol{B}) \tag{4}$$

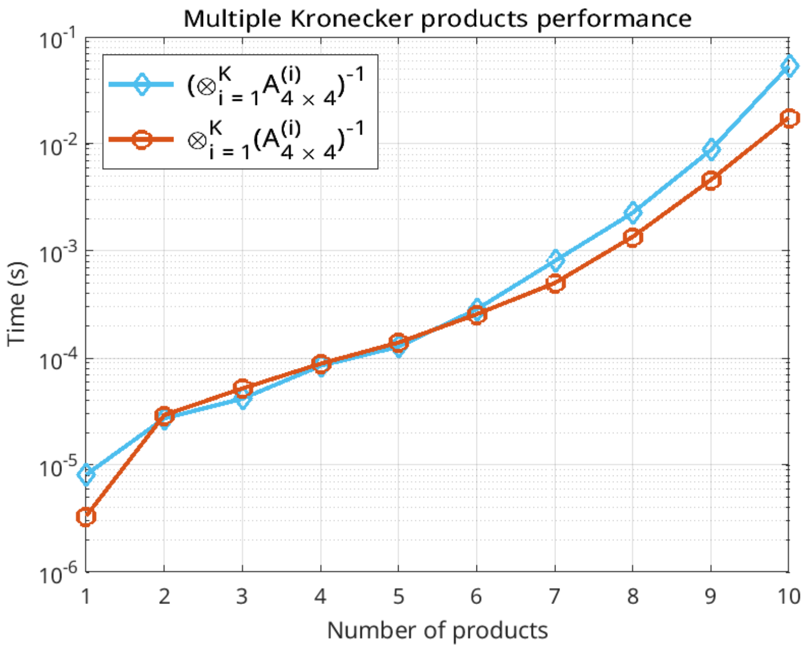

Figure 2: Monter Carlo Experiment with 10000 runs.

Homework 1
Hadamard, Kronecker and Khatri-Rao Products

Run Time Perfomance of Hadamard Product

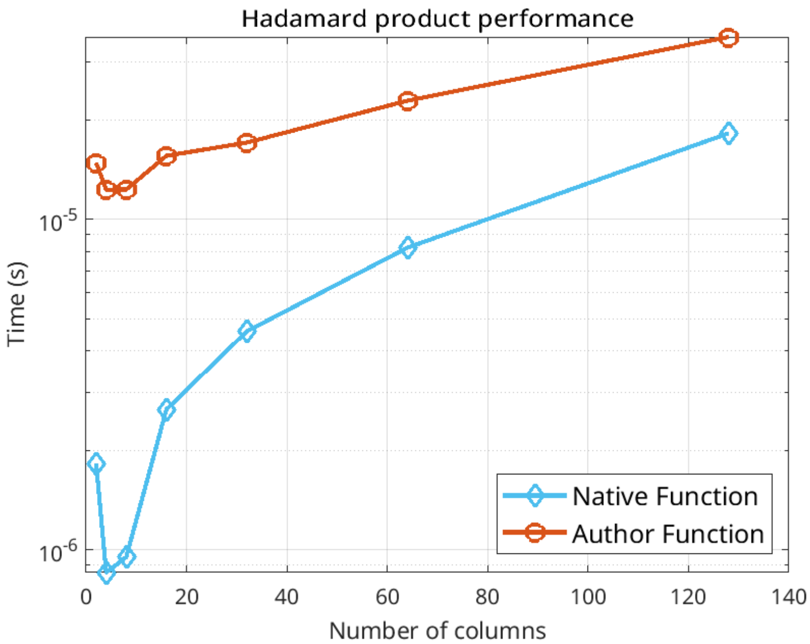

Figure 3: Monter Carlo Experiment with 1000 runs.

Run Time Perfomance of Kronecker Product

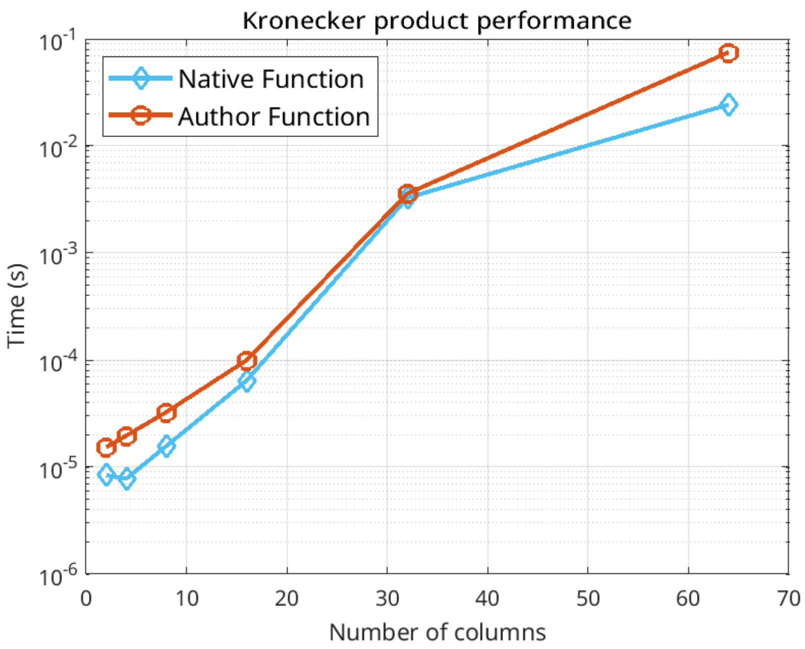

Figure 4: Monter Carlo Experiment with 1000 runs.

Run Time Perfomance of Khatri-Rao Product

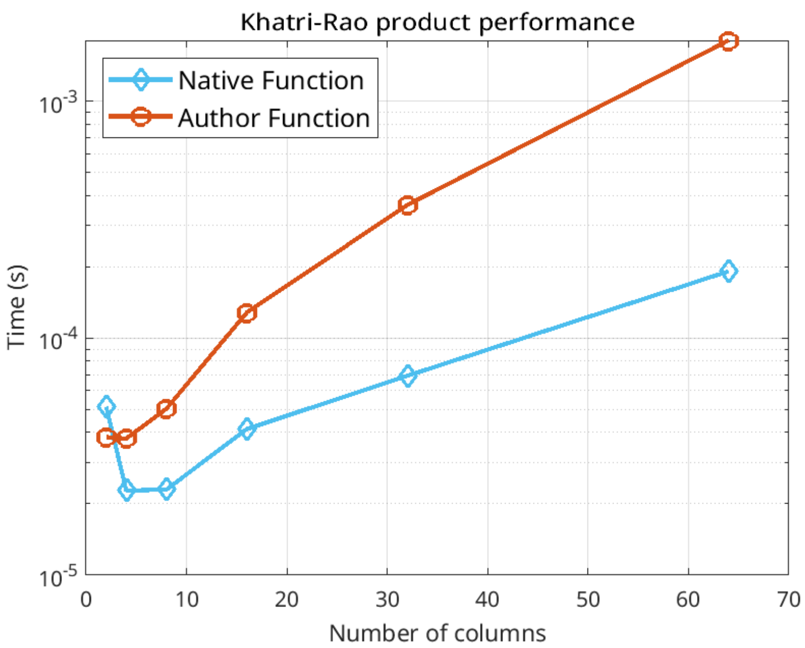

Figure 5: Monter Carlo Experiment with 1000 runs.

Homework 2
Khatri-Rao Product Run Time

Run Time Performance of Khatri-Rao Product for Different Implementations

In Figures 6 and 7 we can see the processing time for the considered methods to compute the pseudoinverse of a Khatri-Rao product for the cases where we have two and four columns in each matrix. To draw the curves it was implemented a Monte Carlo Experiment with only 250 runs for each value N of rows with $N \in \{2, 4, 8, 16, 32, 64\}$. For both cases we can see a clear advantage in using the third method as the dimmensions of the matrices increases. In a similar maner, we can also observe that the second method is a bit better than the first however not as good as the third. To see a better behavior for these methods it should be necessary to increase the number of rows to better analyse the advantages of each one, however due to technical constraints I could not increase as much as I wanted.

$$(\boldsymbol{A} \diamond \boldsymbol{B})^{\dagger} = \mathrm{pinv}(\boldsymbol{A} \diamond \boldsymbol{B}), \tag{5}$$
$$(\boldsymbol{A} \diamond \boldsymbol{B})^{\dagger} = [(\boldsymbol{A} \diamond \boldsymbol{B})^{\mathrm{T}}(\boldsymbol{A} \diamond \boldsymbol{B})]^{-1}(\boldsymbol{A} \diamond \boldsymbol{B})^{\mathrm{T}}, \tag{6}$$
$$(\boldsymbol{A} \diamond \boldsymbol{B})^{\dagger} = [(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B})]^{-1}(\boldsymbol{A} \diamond \boldsymbol{B})^{\mathrm{T}}, \tag{7}$$

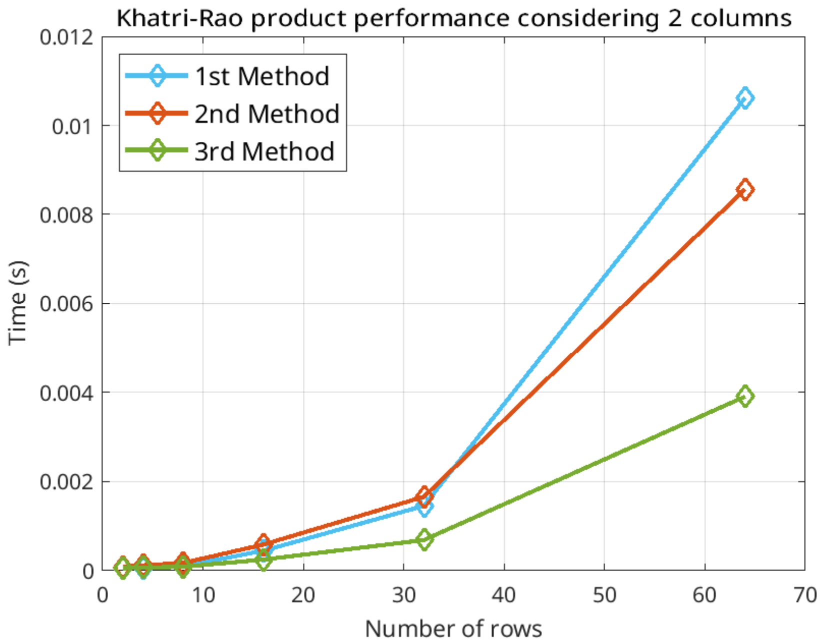

Figure 6: Monter Carlo Experiment with 250 runs and R = 2.

Run Time Perfomance of Sequential Khatri-Rao Products

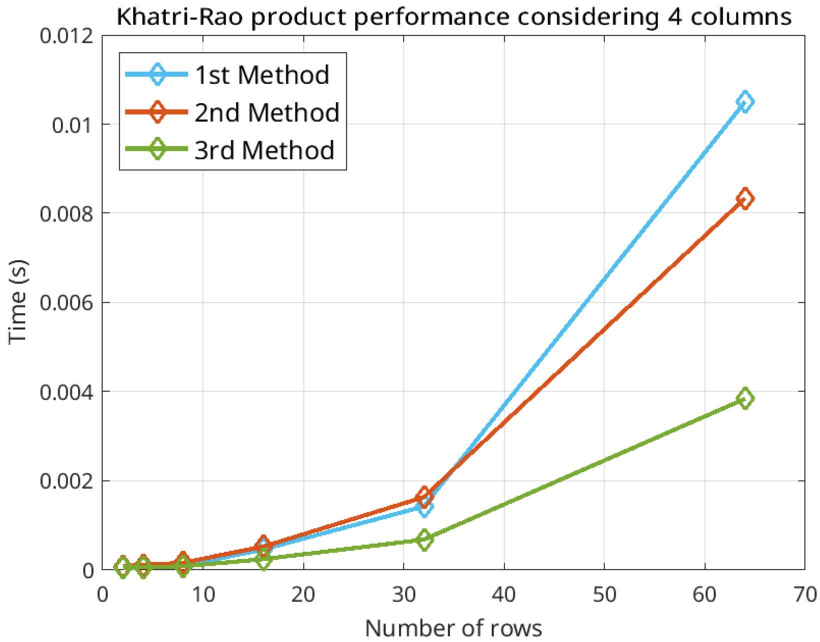

Figure 7: Monter Carlo Experiment with 250 runs and R = 4.

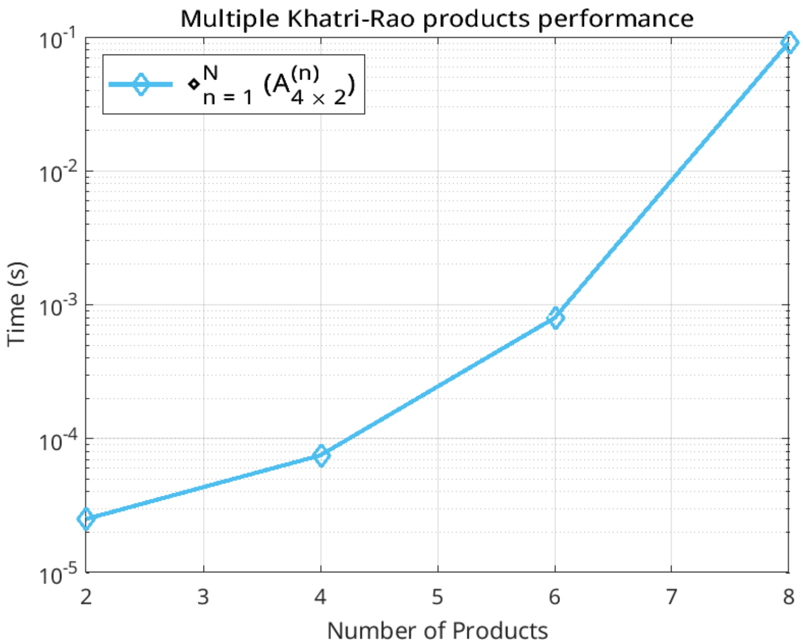

Figure 8: Monter Carlo Experiment with 250 runs.

Homework 3
Least-Squares Khatri-Rao Factorization (LSKRF)

Implementation LSKRF

$$\left(\hat{\boldsymbol{A}}, \hat{\boldsymbol{B}}\right) = \min_{\boldsymbol{A},\boldsymbol{B}} ||\boldsymbol{X} - \boldsymbol{A} \diamond \boldsymbol{B}||_{\mathrm{F}}^{2}, \tag{8}$$

NMSE($\boldsymbol{X}, \hat{\boldsymbol{X}}$)	NMSE($\boldsymbol{A}, \hat{\boldsymbol{A}}$)	NMSE($\boldsymbol{B}, \hat{\boldsymbol{B}}$)
-623.4093	+11.5658	+7.8479

Monte Carlo Experiment

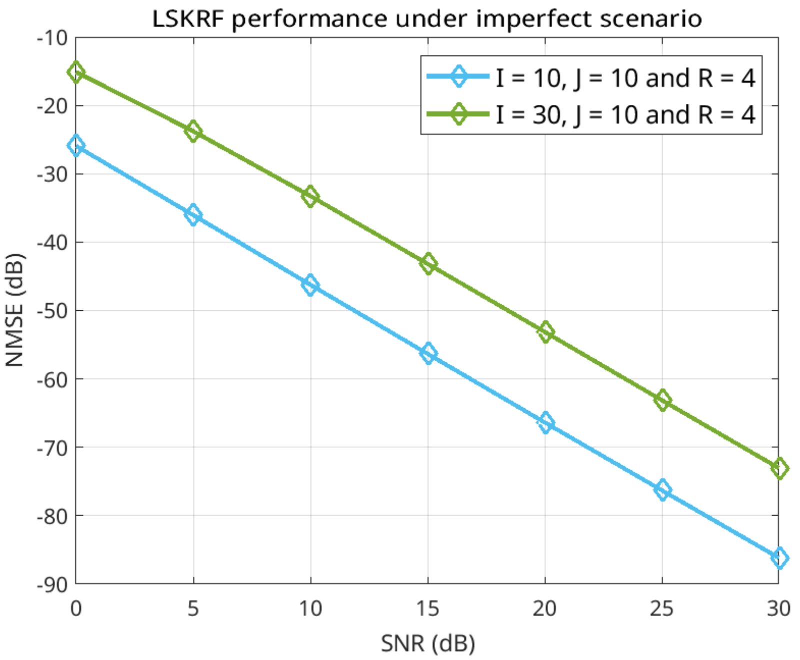

Figure 9: Monter Carlo Experiment with 1000 runs for LSKRF algorithm.

Homework 4
Least Squares Kronecker Product Factorization (LSKronF)

Implementation LSKronF

$$\left(\hat{\boldsymbol{A}}, \hat{\boldsymbol{B}}\right) = \min_{\boldsymbol{A},\boldsymbol{B}} ||\boldsymbol{X} - \boldsymbol{A} \otimes \boldsymbol{B}||_{\mathrm{F}}^2 , \tag{9}$$

NMSE($\boldsymbol{X}, \hat{\boldsymbol{X}}$)	NMSE($\boldsymbol{A}, \hat{\boldsymbol{A}}$)	NMSE($\boldsymbol{B}, \hat{\boldsymbol{B}}$)
-619.2196	+13.5472	+9.5922

Monte Carlo Experiment

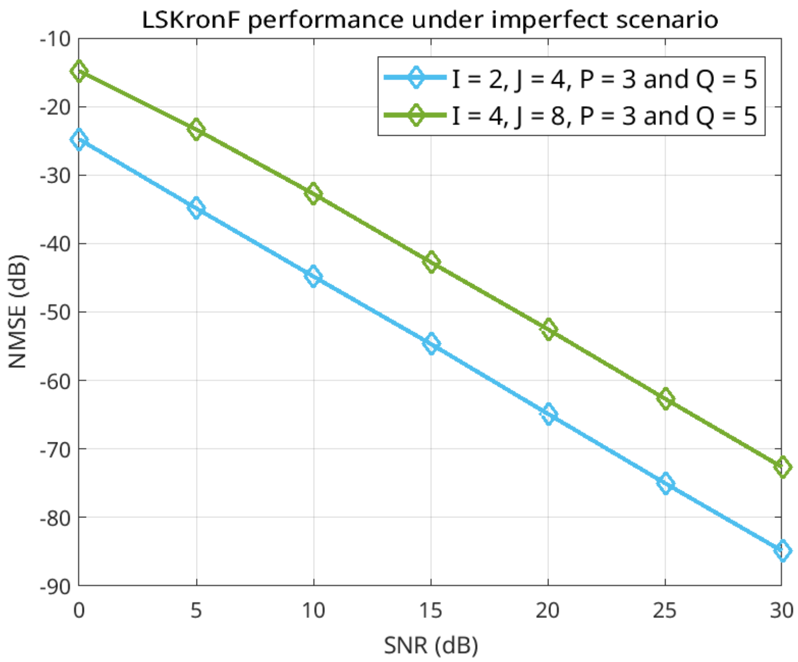

Figure 10: Monter Carlo Experiment with 1000 runs for LSKronf algorithm.

Homework 5
Kronecker Product Singular Value Decomposition (KPSVD)

Implementation KPSVD

$$\boldsymbol{X} = \sum_{k=1}^{r_{kp}} \sigma_k \boldsymbol{U}_k \otimes \boldsymbol{V}_k, \tag{10}$$

In the example is considered that the original matrix has a full rank equals to 9, then two approximations using the KPSVD are provided: One using the full-rank approximation and other using a r-rank approximation ($r < 9$). The NMSE between some approximations and its original matrix are provided in the sequence

Full Rank	Rank = 7	Rank = 5	Rank = 3	Rank = 1
-604.8023	-51.3722	-26.7589	-16.9054	-6.3068

Validation of KPSVD

Homework 6
Unfolding, folding, and n-mode product

Implementation unfolding, folding and n-mode product

$$[\mathcal{X}]_n = \text{unfold}(\mathcal{X}, [I_1 \cdots I_N], n) \in \mathbb{C}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_N}, \tag{11}$$

$$\mathcal{X} = \text{fold}([\mathcal{X}]_n, [I_1 \cdots I_N], n) \in \mathbb{C}^{I_1 \times \cdots \times I_N}, \tag{12}$$

$$\mathcal{Y} = \mathcal{X} \times_1 \boldsymbol{U}_1 \times_2 \cdots \times_N \boldsymbol{U}_N, \tag{13}$$

Validation of unfolding, folding and n-mode product

Homework 7
High Order Singular Value Decomposition (HOSVD)

Implementation HOSVD

NMSE($\mathcal{X}, \hat{\mathcal{X}}$)	NMSE($\mathcal{S}, \hat{\mathcal{S}}$)	NMSE($\boldsymbol{U}_1, \hat{\boldsymbol{U}}_1$)	NMSE($\boldsymbol{U}_2, \hat{\boldsymbol{U}}_2$)	NMSE($\boldsymbol{U}_3, \hat{\boldsymbol{U}}_3$)
-611.2162	+7.7656	+2.6667	+2.0000	+1.6063

Validation of HOSVD

The multilinear rank advantage is maximized when it comes to process sparce tensors since the dimmensions can be greatly reduced without losing of too much relevant information by analysing the profile of its multiples unfoldings.

$$\mathcal{X} \in \mathbb{C}^{8\times4\times10} \rightarrow \hat{\mathcal{X}} \in \mathbb{C}^{R_1\times R_2\times R_3}, \tag{14}$$

$$\mathcal{X} \in \mathbb{C}^{5\times5\times5} \rightarrow \hat{\mathcal{Y}} \in \mathbb{C}^{P_1\times P_2\times P_3}, \tag{15}$$

Homework 8
High Order Order Orthogonal Iteration (HOOI)

Implementation HOOI

NMSE($\mathcal{X}, \hat{\mathcal{X}}$)	NMSE($\mathcal{S}, \hat{\mathcal{S}}$)	NMSE($\boldsymbol{U}_1, \hat{\boldsymbol{U}}_1$)	NMSE($\boldsymbol{U}_2, \hat{\boldsymbol{U}}_2$)	NMSE($\boldsymbol{U}_3, \hat{\boldsymbol{U}}_3$)
-607.9515	+7.2483	+2.6667	+3.9622	+1.16160

Validation of HOOI

The multilinear rank advantage is maximized when it comes to process sparce tensors since the dimmensions can be greatly reduced without losing of too much relevant information by analysing the profile of its multiples unfoldings.

$$\mathcal{X} \in \mathbb{C}^{8\times 4\times 10} \rightarrow \hat{\mathcal{X}} \in \mathbb{C}^{R_1\times R_2\times R_3}, \tag{16}$$

$$\mathcal{X} \in \mathbb{C}^{5\times 5\times 5} \rightarrow \hat{\mathcal{Y}} \in \mathbb{C}^{P_1\times P_2\times P_3}, \tag{17}$$

Homework 9
Multidimensional Least-Squares Khatri-Rao Factorization (MLS-KRF)

Implementation MLS-KRF

$$\left(\hat{\boldsymbol{A}}^{(1)}, \cdots, \hat{\boldsymbol{A}}^{(N)}\right) = \min_{\boldsymbol{A}^{(1)}, \cdots, \boldsymbol{A}^{(N)}} \left|\left|\boldsymbol{X} - \boldsymbol{A}^{(1)} \diamond \cdots \diamond \boldsymbol{A}^{(N)}\right|\right|_{\mathrm{F}}^{2}, \tag{18}$$

NMSE($\boldsymbol{X}, \hat{\boldsymbol{X}}$)	NMSE($\boldsymbol{A}_1, \hat{\boldsymbol{A}}_1$)	NMSE($\boldsymbol{A}_2, \hat{\boldsymbol{A}}_2$)	NMSE($\boldsymbol{A}_3, \hat{\boldsymbol{A}}_3$)
-606.2255	+3.1432	+5.0165	+4.7297

Monte Carlo Experiment

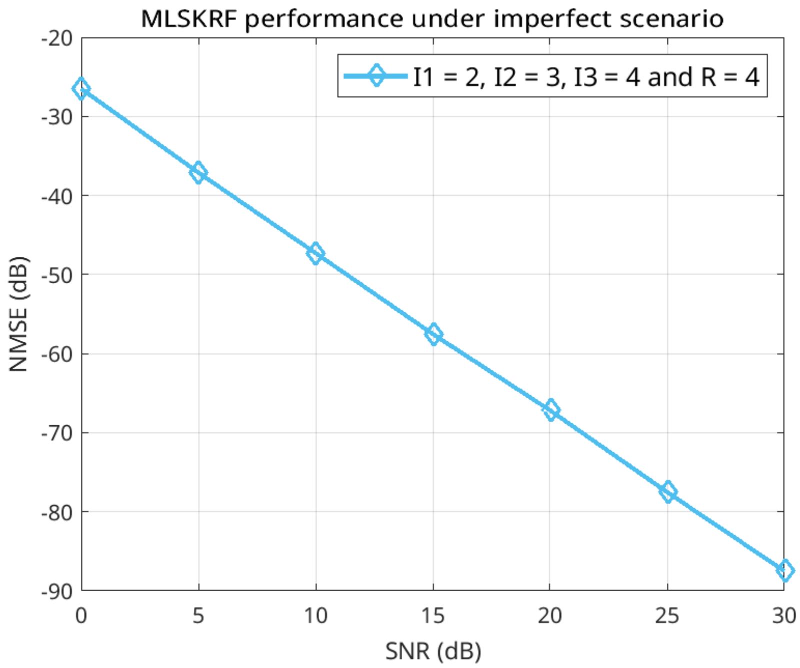

Figure 11: Monter Carlo Experiment with 1000 runs for MLS-KRF algorithm.

Homework 10
Multidimensional Least-Squares Kronecker Factorization (MLS-KronF)

Implementation MLS-KronF

$$\left(\hat{\boldsymbol{A}}^{(1)}, \cdots, \hat{\boldsymbol{A}}^{(N)}\right) = \min_{\boldsymbol{A}^{(1)}, \cdots, \boldsymbol{A}^{(N)}} \left\| \boldsymbol{X} - \boldsymbol{A}^{(1)} \otimes \cdots \otimes \boldsymbol{A}^{(N)} \right\|_{\mathrm{F}}^{2}, \tag{19}$$

NNMSE($\boldsymbol{X}, \hat{\boldsymbol{X}}$)	NMSE($\boldsymbol{A}_1, \hat{\boldsymbol{A}}_1$)	NMSE($\boldsymbol{A}_2, \hat{\boldsymbol{A}}_2$)	NMSE($\boldsymbol{A}_3, \hat{\boldsymbol{A}}_3$)
-605.1941	+11.9214	+11.5548	+6.0950

Monte Carlo Experiment

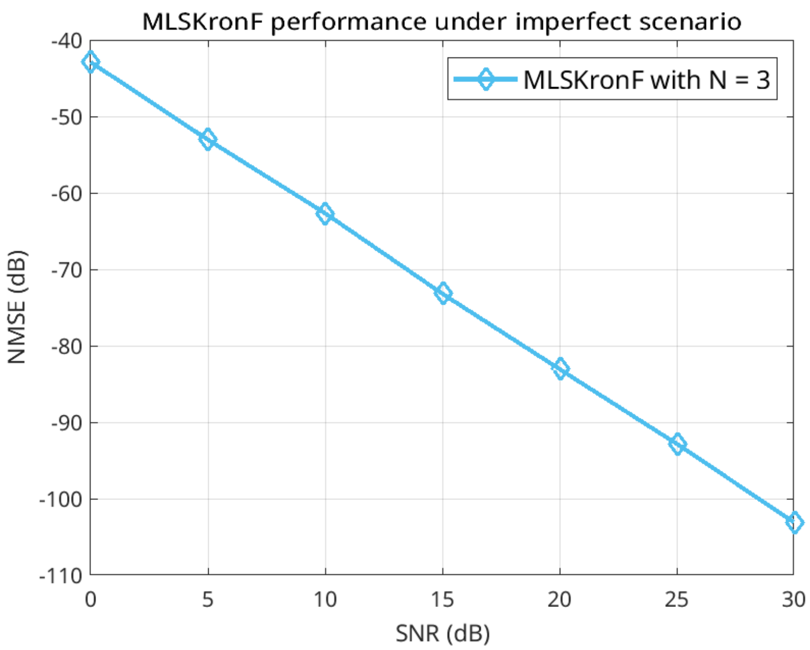

Figure 12: Monter Carlo Experiment with 1000 runs for MLS-KronF algorithm.

Homework 11
Alternating Least Squares (ALS) Algorithm

Implementation of ALS

$$\left(\hat{\boldsymbol{A}}, \hat{\boldsymbol{B}}, \hat{\boldsymbol{C}}\right) = \min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}} \left\| \mathcal{X} - \sum_{r=1}^{R} \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r \right\|_{\mathrm{F}}^{2}, \tag{20}$$

Monte Carlo Experiment

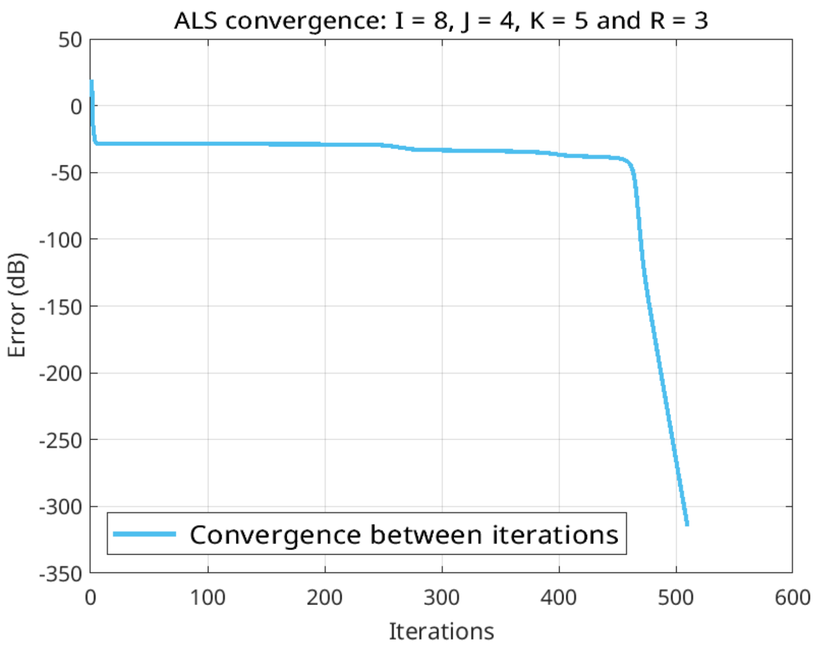

Figure 13: Convergence behavior of ALS algorithm.

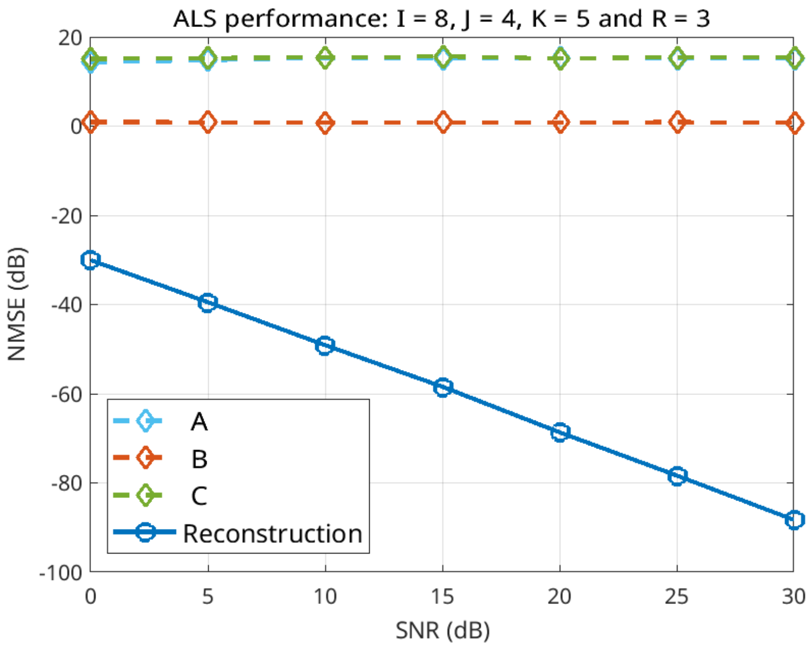

Figure 14: Monter Carlo Experimento with 1000 runs for ALS algorithm.

Homework 12

Tensor Kronecker Product Single Value Decomposition (TKPSVD)

Implementation of TKPSVD

$$\mathcal{X} = \sum_{j=1}^{R} \sigma_i \mathcal{A}_j^{(d)} \otimes \mathcal{A}_j^{(d-1)} \otimes \cdots \otimes \mathcal{A}_j^{(1)}, \tag{21}$$

NMSE($\mathcal{X}, \hat{\mathcal{X}}$)	NMSE($\mathcal{A}^{(1)}, \hat{\mathcal{A}}^{(1)}$)	NMSE($\mathcal{A}^{(2)}, \hat{\mathcal{A}}^{(2)}$)	NMSE($\mathcal{A}^{(3)}, \hat{\mathcal{A}}^{(3)}$)
-625.5192	+37.0803	+0.1706	+32.5731

Monte Carlo Experiment

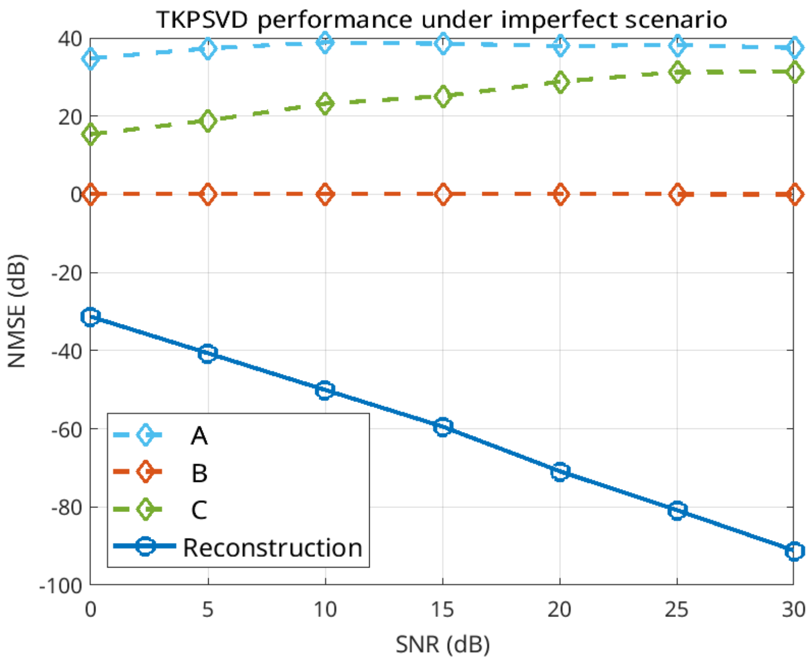

Figure 15: Monter Carlo Experiment with 1000 runs for TKPSVD algorithm.

Homework 13
Tensor Train Single Value Decomposition (TTSVD)

Implementation of TTSVD

$$\mathcal{X} = \boldsymbol{G}_1 \bullet_2^1 \mathcal{G}_2 \bullet_3^1 \mathcal{G}_3 \bullet_4^1 \boldsymbol{G}_4, \tag{22}$$

$$\left(\hat{\boldsymbol{G}}_1, \hat{\mathcal{G}}_2, \hat{\mathcal{G}}_3, \hat{\boldsymbol{G}}_4\right) = \min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}} ||\mathcal{X} - \boldsymbol{G}_1 \bullet_2^1 \mathcal{G}_2 \bullet_3^1 \mathcal{G}_3 \bullet_4^1 \boldsymbol{G}_4||_{\text{F}}, \tag{23}$$

NMSE($\mathcal{X}, \hat{\mathcal{X}}$) with $R = (3, 3, 3)$	NMSE($\mathcal{X}, \hat{\mathcal{X}}$) with $R = (2, 2, 2)$
-606.0326	-25.1084

Monte Carlo Experiment

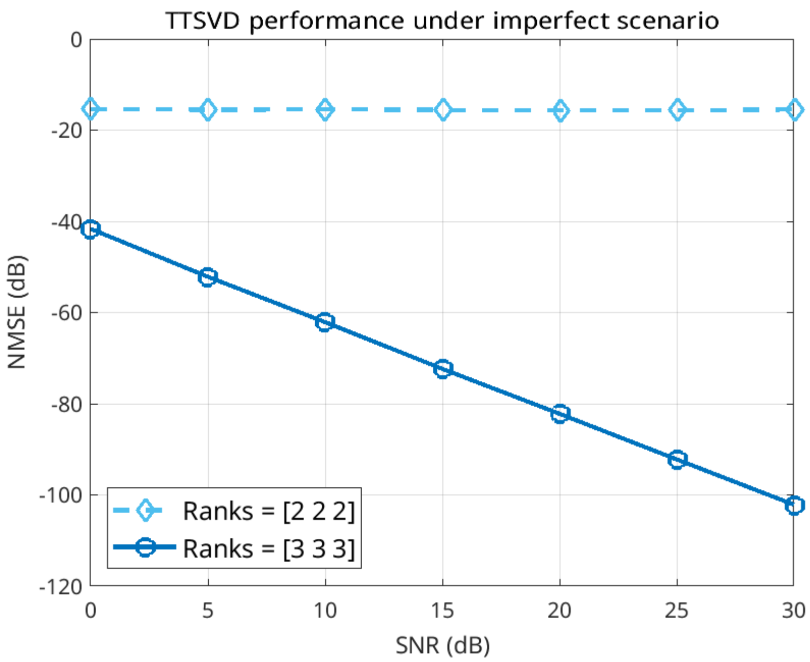

Figure 16: Monter Carlo Experiment with 1000 runs for TTSVD algorithm.