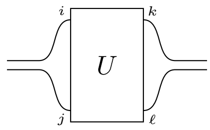
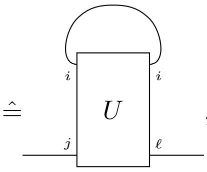
Christian B. Mendl, Richard M. Milbradt due: xx May 20xx (before tutorial)

Exercise 5.1 (Partial trace)
Let $U \in \mathbb{C}^{m\times n\times m\times n}$ be a tensor of degree 4, which we can interpret as $mn \times mn$ matrix by combining the first two and last two dimensions:

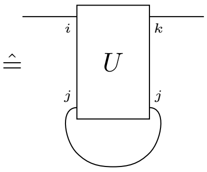


The *partial trace* is defined by "tracing out" (contracting) the subsystem corresponding to the upper two or lower two legs of U:

$$(\mathrm{tr}_1[U])_{j\ell} = \sum_{i=1}^{m} U_{ij,i\ell} \mathrel{\hat{=}}$$



$$, \quad (\mathrm{tr}_2[U])_{ik} = \sum_{j=1}^{n} U_{ij,kj} \mathrel{\hat{=}}$$

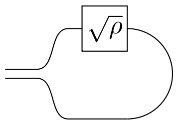


Note that the partial trace yields a matrix: $\mathrm{tr}_1[U] \in \mathbb{C}^{n\times n}$, $\mathrm{tr}_2[U] \in \mathbb{C}^{m\times m}$.

(a) Show graphically that $\mathrm{tr}_1[A \otimes B] = \mathrm{tr}[A]B$ and $\mathrm{tr}_2[A \otimes B] = \mathrm{tr}[B]A$ for any square matrices A and B.

(b) Let $\rho \in \mathbb{C}^{n\times n}$ be a Hermitian, positive semidefinite matrix, such that its eigenvalues are non-negative and $\sqrt{\rho}$ is well defined. Let $\psi \in \mathbb{C}^{n^2}$ be the vectorization of $\sqrt{\rho}$:

$$\psi =$$



Show that $\mathrm{tr}_2[\psi \circ \psi^*] = \rho$.

Hint: $\psi \circ \psi^*$ plays the role of the above U interpreted as matrix. $\sqrt{\rho}$ inherits the Hermitian property from ρ.

Remark: In terms of quantum mechanics, ρ is a "density matrix" and ψ denoted "purification" of ρ. The outer product $\psi \circ \psi^*$ is written in bra-ket notation as $|\psi\rangle\langle\psi|$.

(c) Conversely, let $\psi \in \mathbb{C}^{n^2}$ be an arbitrary vector. Show that $\mathrm{tr}_2[\psi\circ\psi^*]$ is Hermitian and positive semidefinite.

Hint: Revisit the proof of the SVD from the lecture to verify that $AA^\dagger$ is positive semidefinite for any matrix A.

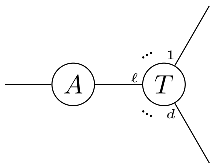
(Continued on page 2)

Exercise 5.2 (Higher-order singular value decomposition)
Before discussing the main algorithm for this exercise, we first require some definitions. Let $T \in \mathbb{C}^{n_1 \times \cdots \times n_d}$ be a tensor of degree d. Then the *ℓ-mode matricization* $T^{(\ell)} \in \mathbb{C}^{n_\ell \times (n_1 \cdots n_{\ell-1} \cdot n_{\ell+1} \cdots n_d)}$ is the partitioning of the dimensions into the ℓ-th dimension of T ($\ell \in \{1, \dots, d\}$) and all the remaining dimensions. The *mulilinear rank* of T is the tuple $(r_1, \dots, r_d)$ with $r_\ell = \operatorname{rank}(T^{(\ell)})$ for $\ell = 1, \dots, d$.

(a) Write a Python function which returns the ℓ-mode matricization of an input tensor T stored as NumPy `array`, with ℓ specified as one of the input arguments of the function.

Hint: First "transpose" (permute dimensions) of T such that the ℓ-th dimension is moved to the front, then reshape the result into a matrix. In accordance with the Python/NumPy convention, ℓ should be zero-based in the code.

For given $\ell \in \{1, \dots, d\}$ and a matrix $A \in \mathbb{C}^{m \times n_\ell}$, the *$\ell$-mode matrix product* $A \cdot_\ell T$ is the multiplication of A with the ℓ-th dimension of T, graphically represented as:

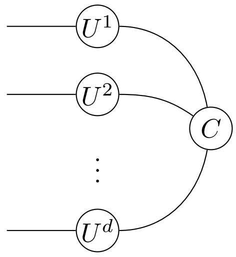
$A \cdot_\ell T \quad \hat{=}$ 

The ordering of the dimensions of T is preserved, thus $A \cdot_\ell T \in \mathbb{C}^{n_1 \times \cdots n_{\ell-1} \times m \times n_{\ell+1} \times \cdots n_d}$.

(b) Implement the ℓ-mode matrix product between a matrix A and tensor T as Python function.

Hint: The NumPy function `tensordot` might be useful for this purpose. Note that you still have to permute the dimensions of the output of `tensordot`, to restore the original ordering from T.

Now recall the *Tucker format*, which is a particular decomposition of a tensor as



with $\quad U^\ell \in \mathbb{C}^{n_\ell \times k_\ell}$ for $\ell = 1, \dots, d, \quad C \in \mathbb{C}^{k_1 \times \cdots \times k_d}$.

(c) Explain why the multilinear rank of such a Tucker format tensor is (pointwise) at most $(k_1, \dots, k_d)$.

Hint: You can use without proof that $\operatorname{rank}(AB) \le \min(m, k, n)$ for any $A \in \mathbb{C}^{m \times k}$ and $B \in \mathbb{C}^{k \times n}$.

The higher-order singular value decomposition (HOSVD) allows to approximate a given tensor T by another tensor in Tucker format (i.e., find corresponding U^ℓ matrices and the C tensor), with prescribed dimensions $k_1, \dots, k_d$. These dimensions play the role of the number of retained singular values in the conventional low-rank approximation of a matrix by SVD. As pseudocode:

HOSVD algorithm

Input: tensor $T \in \mathbb{C}^{n_1 \times \cdots \times n_d}$, tuple $(k_1, \dots, k_d)$ with $k_\ell \le n_\ell$ for all ℓ

1. Compute the SVDs of the ℓ-mode matricizations: $T^{(\ell)} = U^\ell S^\ell (V^\ell)^\dagger$ for $\ell = 1, \dots, d$, with $S^\ell = \operatorname{diag}(\sigma^\ell)$, $\sigma^\ell = (\sigma_1^\ell, \sigma_2^\ell, \dots)$
2. Truncate: $\tilde{U}^\ell = U^\ell_{:,1:k_\ell}$ (first k_ℓ columns of U^ℓ) for $\ell = 1, \dots, d$
3. Form the core tensor $\tilde{C}$:
   Initialize $\tilde{C} \leftarrow T$
   `for` $\ell = 1, \dots, d$:
   $\quad\tilde{C} \leftarrow (\tilde{U}^\ell)^\dagger \cdot_\ell \tilde{C}$ (project onto the subspace spanned by the columns of $\tilde{U}^\ell$)

`return` $(\tilde{U}^1, \dots, \tilde{U}^d, \tilde{C}, \sigma^1, \dots, \sigma^d)$, which defines the Tucker approximation of T

The returned singular values are not strictly necessary, but useful as diagnostic information.

(d) Implement the HOSVD algorithm, using the two functions from (a) and (b).

Hint: For step 1 of the algorithm, use `np.linalg.svd` with the parameter `full_matrices=False`.

(e) The Moodle page contains data from a computations fluid dynamics simulation (two spatial and one time dimension), and a Jupyter notebook template for this exercise. Complete and run the notebook using your implementations of (a), (b) and (d).