Guidelines

- Write your name and matriculation number on each sheet of paper.
- Only clearly readable exercise-elaborations are evaluated.
- Results have to be provided together with an evident way of calculation.
- Keep textual answers short and concise. Lengthy or vague statements won't gain points.

Exercise 8.1 (0.5 points)
In this exercise we analyze the $N \times N$ matrix:

$$\bar{I}_N = \begin{bmatrix} 0 & 1 & \cdots & \cdots & 1 \\ 1 & 0 & 1 & \dots & 1 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & \ddots & \ddots & 0 & 1 \\ 1 & \cdots & \cdots & 1 & 0 \end{bmatrix}$$

1. Calculate the eigenvalues of
   - $\bar{I}_N$
   - $I_N \otimes \bar{I}_N$
   - $\bar{I}_N \otimes I_N$
   - $M_N = I_N \otimes \bar{I}_N + \bar{I}_N \otimes I_N$
2. Calculate the eigenvalues of the terms
   - $wI_{N^2} + \frac{1-w}{2} M_N$
   - $\frac{I_{N^2} - wM_N}{1+2w}$

   in dependence of w. Assume $N > 1$ and calculate that w which gives you a contraction mapping (i.e., the absolute value of all eigenvalues is less than 1).

Terms: Kronecker product, eigenvalues, eigenvectors.

Exercise 8.2 (0.5 points)
In this example, we will investigate the differences between a Discrete Fourier Transform (DFT) and the Fast Fourier Transform (FFT) algorithm, based on the Kronecker product.

1. Consider a DFT of length $N = 6$. Explicitly state the input/output relation for the DFT
$$\underline{y} = F_6 \underline{x}$$
and for the FFT algortihm as
$$\underline{\tilde{y}} = F_x \underline{\tilde{x}}, \quad F_x = F_3 \otimes F_2$$
with the elements of $\underline{\tilde{y}}, \underline{\tilde{x}}$ in the appropriate order.
2. Sketch a signal flow diagram of the FFT algorithm.
3. Compare the complexity of the standard DFT to the FFT algorithm. (How many mult-add operations are necessary?)
4. Now consider a DFT/FFT of length 72. Again, compare the complexity.

Terms: Discrete Fourier Transform, Fast Fourier Transform, Kronecker product.

Exercise 8.3 (0.5 points)
Let us first consider the *rank decomposition* of a matrix. The rank of an I × J matrix X is the smallest number of rank one matrices (vector outer products of the form $\underline{a} \circ \underline{b} = \underline{a}\underline{b}^T$) needed to synthesize X as

$$X = \sum_{r=1}^{\mathrm{R}} \underline{a}_r \circ \underline{b}_r = AB^T. \tag{1}$$

1. What are A and B with respect to $\underline{a}_r$ and $\underline{b}_r$, respectively?
   How do you express matrix element $(X)_{i,j}$ in terms of the elements in $\underline{a}$, $\underline{b}$?

Consider now the matrices

$$H_1 = \begin{bmatrix} 2 & 4 \\ 0 & 0 \\ 1 & 2 \end{bmatrix}, \qquad H_2 = \begin{bmatrix} 2 & 4 \\ 3 & 1 \\ 1 & 2 \end{bmatrix}. \tag{2}$$

2. Show that H_1 has rank 1. Given $\underline{b}_1 = [1,\, 2]^T$, compute $\underline{a}_1$ that satisfies Equation (1) with $X = H_1$.

3. Show that H_2 has rank 2. Assume that the rank decomposition of H_2 has the same $\underline{a}_1$ and $\underline{b}_1$ as H_1. Furthermore, assume that $\|\underline{a}_2\|_2 = 1$ and compute the $\underline{b}_2$ that satisfies Equation (1) with $X = H_2$.

Similarly to matrices, the rank of an I × J × K three-way array (tensor) $\underline{X}$ is the smallest number of outer products needed to synthesize $\underline{X}$ as

$$\underline{X} = \sum_{r=1}^{\mathrm{R}} \underline{a}_r \circ \underline{b}_r \circ \underline{c}_r. \tag{3}$$

4. How do you express array element $(\underline{X})_{i,j,k}$ in terms of the elements in $\underline{a}$, $\underline{b}$ and $\underline{c}$?

Such multiway arrays occur frequently in the age of big data. In many applications, it is not possible to store these huge data structures due to memory limitations. Thus, compression is applied that tries to retain the latent features of the data. Figure 8.1 depicts an example where an I × J × K tensor $\underline{X}$ is compressed into a much smaller $\widetilde{\mathrm{I}} \times \widetilde{\mathrm{J}} \times \widetilde{\mathrm{K}}$ tensor $\underline{\widetilde{X}}$ by applying the compression matrices U, V and W.
Assume now a noisy tensor

$$\underline{Y} = \underline{X} + \underline{Z}, \tag{4}$$

where $\underline{Z}$ denotes additive white noise. Let us first investigate under which conditions the noise remains white after compression. The vectorized form of Equation (4) is obtained as $\underline{y} = \underline{x} + \underline{z}$, where $\underline{y} = \mathrm{vec}(\underline{Y})$, $\underline{x} = \mathrm{vec}(\underline{X})$, and $\underline{z} = \mathrm{vec}(\underline{Z})$.

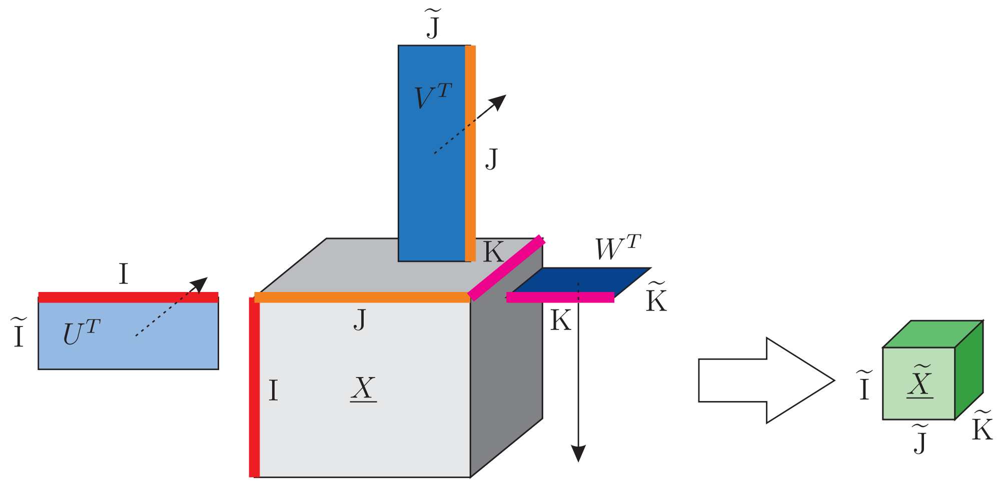

Figure 8.1: Compression of tensor $\underline{X}$ by matrices U, V, and W.

After multiway compression similar to Figure 8.1, we obtain the reduced-size tensor $\underline{\tilde{Y}}$ whose vectorized representation reads

$$\underline{\tilde{y}} = \mathrm{vec}(\underline{\tilde{Y}}) = (U^T \otimes V^T \otimes W^T)\underline{y} = \underbrace{(U^T \otimes V^T \otimes W^T)\underline{x}}_{\underline{\tilde{x}}} + \underbrace{(U^T \otimes V^T \otimes W^T)\underline{z}}_{\underline{\tilde{z}}}.$$

5. Assume white noise with expectation $\mathrm{E}\{\underline{z}\underline{z}^T\} = \sigma^2 I_{\mathrm{IJK}}$ (I_{IJK} is the identity matrix of dimension $\mathrm{I} \cdot \mathrm{J} \cdot \mathrm{K}$). Note that U, V and W are deterministic.

   Show that the expectation of the compressed noise computes as

$$\mathrm{E}\{\underline{\tilde{z}}\underline{\tilde{z}}^T\} = \sigma^2 \left((U^TU) \otimes (V^TV) \otimes (W^TW)\right).$$

   Hint: use mixed product rule $(A \otimes B)(C \otimes D) = (AC \otimes BD)$.

6. For which U, V and W is the compressed noise white, i.e., $\mathrm{E}\{\underline{\tilde{z}}\underline{\tilde{z}}^T\} = I_{\tilde{\mathrm{I}}\tilde{\mathrm{J}}\tilde{\mathrm{K}}}$?

Let us now consider a rank-one tensor $\underline{X} = \underline{a} \circ \underline{b} \circ \underline{c}$ written in vectorized form as $\underline{x} = \underline{a} \otimes \underline{b} \otimes \underline{c}$.

7. Show that $\|\underline{\tilde{x}}\|_2^2 = \|U^T\underline{a}\|_2^2 \cdot \|V^T\underline{b}\|_2^2 \cdot \|W^T\underline{c}\|_2^2$.

   Hint: explicitly compute $\underline{\tilde{x}}$ *first by utilizing the mixed product rule.*

Terms: rank decomposition, tensor, Kronecker product, mixed product rule.

MATLAB-Exercise 8.1 (1 point)
As an extension to analytical example 8.3, we now take a closer look at a *three-way array* (tensor) of *rank one*, i.e.,

$$\underline{X} = \underline{a} \circ \underline{b} \circ \underline{c},$$

where vectors $\underline{a}$, $\underline{b}$ and $\underline{c}$ have dimension I, J and K, respectively. Tensor $\underline{X}$ is thus an array of dimension $\mathrm{I} \times \mathrm{J} \times \mathrm{K}$.

1. Generate $\underline{X}$ with the given vectors $\underline{a}$, $\underline{b}$ and $\underline{c}$ (load `vectors.mat` from the course homepage, the vectors should appear in the workspace). How many nonzero elements does $\underline{X}$ contain?

   Hint: utilize the expression from Exercise 8.3.4.

2. Verify that the vectorized version of tensor $\underline{X}$, i.e., $\mathrm{vec}(\underline{X})$, is equal to the vectorized version obtained by the Kronecker product $\underline{x} = \underline{a} \otimes \underline{b} \otimes \underline{c}$.

   Hint: to generate $\underline{x}$, use function `kron` *and carefully consider the order of the Kronecker products in Matlab.*

3. Assume that vector $\underline{b}$ and tensor $\underline{X}$ are given and that you know that all nonzero entries in $\underline{c}$ have value 1. Compute vectors $\underline{a}$ and $\underline{c}$.

   Hint: work on matrices (slabs of tensor $\underline{X}$) and utilize the method from Exercise 8.3.2.

We will now consider the compression of three-way array (tensor) $\underline{X}$ with matrices U, V and W as illustrated by Figure 8.1 in Exercise 8.3. The compression matrices $U \in \{-1{,}1\}^{\mathrm{I}\times\widetilde{\mathrm{I}}}$, $V \in \{-1{,}1\}^{\mathrm{J}\times\widetilde{\mathrm{J}}}$ and $W \in \{-1{,}1\}^{\mathrm{K}\times\widetilde{\mathrm{K}}}$ can be found on the course homepage (load `compmat.mat`).

4. Perform compression on three-way array $\underline{X}$. Apply the compression matrices on slabs (matrices) of $\underline{X}$, see Figure 8.1. Be careful to iterate over the correct dimension (slabs). Note that after the compression of each dimension (I, J and K), the resulting array is changed in size. Ultimately, the compressed three-way array $\underline{\widetilde{X}}$ has dimension $\widetilde{\mathrm{I}} \times \widetilde{\mathrm{J}} \times \widetilde{\mathrm{K}}$.

   Hint: to copy the elements of a matrix `B` *into a slab of a three-way array* `A`*, e.g.* `A(i,:,:) = B`*, be careful about the dimensions in* `A`*. In this example, the first dimension of* `A` *is a singleton dimension, and* `B` *has to be augmented to fit the dimensions. Helpful functions to do so:* `reshape, shiftdim`*. Similarly, to assign a three-way array slab to a matrix, e.g.* `B = A(:,j,:)`*, you have to remove the singleton dimension using e.g.* `squeeze`*.*

5. Perform compression on the vectorized representation of $\underline{X}$ according to

$$\underline{\widetilde{x}} = (U^T \otimes V^T \otimes W^T)\underline{x} = (U^T \otimes V^T \otimes W^T)(\underline{a} \otimes \underline{b} \otimes \underline{c}) = \underline{\widetilde{a}} \otimes \underline{\widetilde{b}} \otimes \underline{\widetilde{c}}$$

and verify that $\mathrm{vec}(\underline{\widetilde{X}}) = \underline{\widetilde{x}}$.

Hint: first, figure out $\underline{\widetilde{a}}$, $\underline{\widetilde{b}}$ and $\underline{\widetilde{c}}$ using the mixed product rule and then generate $\underline{\widetilde{x}}$ in Matlab. Again, be careful about the element ordering using `kron`*.*

Remember the compressed sensing example from Exercise 7. If our rank-one tensor $\underline{X}$ features latent sparsity, i.e., the vectors $\underline{a}$, $\underline{b}$ and $\underline{c}$ are sparse and contain only few nonzero elements, the compression of each dimension can be interpreted as a compressed sensing measurement, e.g., $\underline{\widetilde{a}} = U^T\underline{a}$, where $\underline{a}$ is compressed by matrix U^T to obtain compressed representation $\underline{\widetilde{a}}$. It is important to recognize that the compressed three-way array $\underline{\widetilde{X}}$ is also a rank one tensor that can be synthesized by the three vectors $\underline{\widetilde{a}}$, $\underline{\widetilde{b}}$ and $\underline{\widetilde{c}}$. This makes reconstruction of $\underline{X}$ feasible, since we can reconstruct $\underline{a}$, $\underline{b}$ and $\underline{c}$ that synthesize $\underline{X}$ from the compressed versions $\underline{\widetilde{a}}$, $\underline{\widetilde{b}}$ and $\underline{\widetilde{c}}$ that synthesize $\underline{\widetilde{X}}$.
If the synthesis is not known and we want to reconstruct (vectorized tensor) $\underline{x}$ from

$$\underline{\widetilde{x}} = \underbrace{(U^T \otimes V^T \otimes W^T)}_{S}\,\underline{x},$$

we run into trouble as the dimension of sensing matrix S is huge (try to generate S with the given matrices, Matlab will prompt out of memory).
It is thus important to exploit tensor features such as low-rank and sparsity.

Terms: three-way array, tensor, Kronecker product, compression, sparsity