Christian B. Mendl, Richard M. Milbradt due: 12 May 2022 (before tutorial)

Tutorial 2 (Spectral and singular value decomposition)
We consider the matrix

$$A = \begin{pmatrix} 1 & -i & 0 \\ i & 1 & 0 \\ 0 & 0 & i \end{pmatrix}.$$

(a) Show that A is normal, and compute its characteristic polynomial and spectral decomposition.

(b) Compute the singular values of A. What is the rank of A?

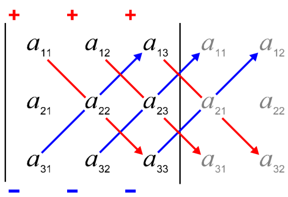
(c) Let B be a unitary matrix. Provide a singular value decomposition of B.
Note: The matrices U and V are not unique.

Exercise 2.1 (Spectral decomposition)

(a) Compute the characteristic polynomial and spectral decomposition of the normal matrix

$$A = \begin{pmatrix} 0 & \frac{3}{5} & \frac{4}{5} \\ -\frac{3}{5} & 0 & 0 \\ -\frac{4}{5} & 0 & 0 \end{pmatrix}.$$

Hint: The following "rule of Sarrus" might be helpful for calculating the determinant of a 3×3 matrix:



Source: Wikipedia

(b) Let $A \in \mathbb{C}^{n\times n}$ be a normal matrix, and $\{\lambda_1, \dots, \lambda_n\}$ the eigenvalues of A. Show that

$$\operatorname{tr}[A] = \sum_{j=1}^{n} \lambda_j,$$

where $\operatorname{tr}[\cdot]$ denotes the matrix trace (sum of its diagonal entries).

Hint: Start from the spectral decomposition of A, and use that $\operatorname{tr}[AB] = \operatorname{tr}[BA]$ for any square matrices A and B.

Exercise 2.2 (Python and NumPy, and SVD compression)
The Python NumPy library (https://numpy.org) is an accessible and powerful tool for linear algebra and tensor operations. It is organized around an `array` type, which supports tensors of arbitrary degree (i.e., vectors, matrices, and higher-degree tensors). Please familiarize yourself with Python/NumPy. We also recommend to use a local installation and development tools like Spyder or Jupyter notebooks.

(For the homework submission, please upload your code and the generated output in a single PDF file.)

(a) Compute the spectral and singular value decomposition of A defined in the above tutorial using NumPy.

Hint: `numpy.linalg.eig` and `numpy.linalg.svd`

(b) The singular value decomposition can be used to "compress" a matrix (approximate it with less data). Specifically, let $A \in \mathbb{C}^{n\times n}$ with associated SVD decomposition $A = USV^\dagger$, where $S = \operatorname{diag}(\sigma_1, \dots, \sigma_n)$ is the diagonal matrix of singular values. Then one can obtain a compressed representation by retaining only the leading χ singular values ($\chi \in \{1, \dots, n\}$), i.e., setting

$$\tilde{U} = U_{:,1:\chi} \text{ (first } \chi \text{ columns of } U), \quad \tilde{S} = \operatorname{diag}(\sigma_1, \dots, \sigma_\chi), \quad \tilde{V} = V_{:,1:\chi} \text{ (first } \chi \text{ columns of } V),$$

such that

$$A \approx \tilde{U}\tilde{S}\tilde{V}^\dagger.$$

χ controls the amount of compression, from no compression at all ($\chi = n$), to approximating A by an outer product of two vectors ($\chi = 1$). Note that $\tilde{U}\tilde{S}\tilde{V}^\dagger$ has rank χ by construction (assuming $\sigma_\chi > 0$). By the Eckart-Young-Mirsky theorem, the above is the best rank-χ approximation of A.

The Moodle page contains a template (as Jupyter notebook) for applying this method to image compression. Complete the tasks there and run the notebook.