Dr. Christian Vollmann *Winter Term 2022*

[February 15, 2023]

Numerical Linear Algebra

all exercises

Due date: –

Name:

Matriculation number:

Remarks

- Upload your solutions to the respective Homework directory on olat:

  Homework/<sheetnumber>/Participant Drop Box

- Follow these guidelines:

	Theory	**Programming**
Identifier	no gray tag "Python"	gray tag "Python"
Group work	no	yes (up to 3, but not mandatory!)
Upload format	.pdf	.ipynb
Filename	`sheetnumber_Name.pdf`	`sheetnumber_Name1Name2Name3.ipynb`

1 Math Basics

Ex 1 Math Basics

Binomial Coefficient The binomial coefficient is defined by

$$\binom{n}{k} = \frac{n!}{k!(n-k)!},$$

where $n! = n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 1$. Pascal's triangle states the equation

$$\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}$$

for any $0 \leq k < n$. Please prove this equation.

Ex 2 Math Basics

Complex Numbers
Let i be the imaginary unit, i.e., $i^2 = -1$. Please cast the following complex numbers into the format $z = x + iy$.

1. $(4 + 5i)(4 - 5i)$
2. $\frac{2+3i}{4+5i}$
3. $\sqrt{16b} + \sqrt{3a - 12a}$

Ex 3

Show by induction that for any $n \in \mathbb{N}$ it holds that

$$\sum_{k=0}^{n} (2k+1) = (n+1)^2.$$

Ex 4

Show by induction that for any $n \in \mathbb{N}$ it holds that

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}.$$

Ex 5

Show by induction that for any $n \in \mathbb{N}$ it holds that

$$\sum_{k=1}^{n} k(k+1) = \frac{1}{3} n(n+1)(n+2).$$

Ex 6

Proof via Induction
Please prove the following assertions by induction.

1. $\sum_{k=0}^{n} (2k+1) = (n+1)^2$
2. $2^n = \sum_{k=0}^{n} \binom{n}{k}$

3. Let $q \neq 1$ then

$$\sum_{k=0}^{n} q^k = \frac{q^{n+1} - 1}{q - 1}.$$

Hint: Use Exercise 12 for (ii).

Ex 7 Math Basics

Solution Sets
Please indicate the solution set of the following inequalities.

1. $\frac{3-x}{2} < 5$
2. $3 - (x+1)^2 \geq 2 + \frac{2}{3}x$
3. $5(x + 3(1-x) + 2) \geq 7$
4. $x + 2 \geq -\frac{1}{x}$

Ex 8 Math Basics

Result for Injective Funtions
Let $f : X \to Y$ be an injective function and $A, B \subset X$. Please show that

$$f(A) \cap f(B) = f(A \cap B).$$

Hint: Split up the equality "$=$" into the parts "$\subset$" and "$\supset$". One direction is straightforward the other one requires, that f is injective.

Ex 9 Math Basics

Irrational $\sqrt{2}$
Please show that $\sqrt{2}$ is not a rational number.

Ex 10 Math Basics

Logical Negation
Please state the negation of the following statements.

1. It rains.
2. It rains and it is cold outside.
3. $A \wedge B$ (*Hint:* De Morgan)
4. $x = 3$ or $x = 1$
5. $A \vee B$
6. If it rains the street gets wet.
7. $A \Rightarrow B$

Ex 11 Math Basics

PQFormula
Let $a, b \in \mathbb{R}$ and $a \neq 0$. Please solve the following quadratic equations.

1. $x^2 + bx - 2b^2 = 0$
2. $3a^2x^2 + 4ax + 1 = 0$
3. $ax^2 + ax + x + 1 = 0$

Ex 12

Heron's algorithm
For a nonnegative number $a \geq 0$ compute the square root $\sqrt{a}$ up to an error of 10^{-10}. For this purpose, use the following iteration:

$$x_{n+1} = \frac{1}{2}\left(\frac{x_n^2 + a}{x_n}\right).$$

Choose different initial values x_0 and observe the convergence behavior by printing the error $|x_n - \sqrt{a}|$ in each iteration step.
Hint: Use a while-loop for the iteration (`while`) and the built-in function abs as well as the Python value a `** 0.5` to compute the error $|x_n - \sqrt{a}|$ in each iteration step.

Ex 13 Math Basics

Set Operations
Let $M = \{\alpha, 3, \gamma\},\ N = \{\square, \gamma, \star, \circ\}$ and $K = \{\alpha, \{\alpha\}, M\}$. Please indicate the sets

1. $M \cap N$,
2. $M \cap K$,
3. $\mathcal{P}(M) \cap K$,
4. $M \setminus K$ and
5. $N \cup K$.

Ex 14 Math Basics

Sets and Functions
Let A, B be finite sets which contain the same number of elements, i.e., $|A| = |B|$ and let $f : A \to B$ be a function. Please show that the following statements are equivalent.

1. f is injective
2. f is surjective
3. f is bijective

Hint: (iii) follows immediately from (i) $\Leftrightarrow$ (ii).

Ex 15 Math Basics

Subset Property
Please show that the following assertions are equivalent.

1. $A \subset B$
2. $A \cap B = A$
3. $A \cup B = B$

Hint: It suffices to show $(i) \Rightarrow (ii) \Rightarrow (iii) \Rightarrow (i)$.

Ex 16 Math Basics

Symmetric Difference
Let X be a set and $A, B \subset X$ be two subsets of X. The symmetric difference of A and B is then defined by

$$A \Delta B := (A \cap B^c) \cup (B \cap A^c).$$

Please show that

$$A \Delta B = (A \cup B) \cap (A \cap B)^c.$$

Hint: Use the De Morgan's rule and exploit the following distributive laws:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \quad \text{and} \quad A \cup (B \cap C) = (A \cup B) \cap (A \cup C).$$

Ex 17 Math Basics

Triangle Inequality
Let $x, y \in \mathbb{R}$. Please show the triangle inequality and the reverse triangle inequality.

1. $|x + y| \leq |x| + |y|$
2. $|x - y| \geq ||x| - |y||$

Ex 18 Math Basics

VennDiagrams
Please draw Venn-Diagrams which illustrate the following statements.

1. (Commutative property)
   $A \cup B = B \cup A$
   $A \cap B = B \cap A$
2. (Associative property)
   $(A \cup B) \cup C = A \cup (B \cup C)$
   $(A \cap B) \cap C = A \cap (B \cap C)$
3. (Distributive property)
   $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$
   $(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$

2 Linear Algebra

Ex 19 Linear Algebra

Copper, Silver and Gold Alloys
Let us assume we are given the alloys M_1, M_2 and M_3, which are built of copper, silver and gold with the following percental proportions:

	M_1	M_2	M_3
Copper	20	70	50
Silver	60	10	50
Gold	20	20	0

Is it possible to mix these alloys to form a new alloy which consists of 40% copper, 50% silver and 10% gold.

Hint: Cast the problem into a linear system $Ax = b$.

Ex 20

Let $A \in \mathbb{R}^{n\times n}$ be matrix for which $A^\top = -A$ (*antisymmetric*). Show that the diagonal entries are zero, i.e., $a_{ii} = 0$ for all $1 \leq i \leq n$.

Ex 21

What is the determinant of

$$A = \begin{pmatrix} 0 & -e^{-i\pi} & i^2 & 0 & -\pi \\ e^{-i\pi} & 0 & -2 & 0 & 0 \\ 1 & 2 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -3 \\ \pi & 0 & 0 & 3 & 0 \end{pmatrix}?$$

Ex 22 Linear Algebra, Determinant

Compute Determinants
Compute the determinants of the following matrices and check whether they are invertible.

$$A = \begin{pmatrix} 1 & \pi & 2 & 12 \\ 0 & \frac{1}{5} & \frac{1}{\sqrt{2}} & 17 \\ 0 & 0 & 5 & \frac{1}{3} \\ 0 & 0 & 0 & 4 \end{pmatrix}, \qquad B = \begin{pmatrix} 5 & 3 \\ 1 & -2 \end{pmatrix}, \qquad C = \begin{pmatrix} 0 & \frac{1}{2} & 2 \\ -\frac{1}{2} & 0 & 7 \\ -2 & -7 & 0 \end{pmatrix}.$$

Ex 23 Linear Algebra

Examples for (skew-)symmetric Matrices
Give examples for symmetric and skew-symmetric (exercise **??**) matrices $A \in \mathbb{R}^{2\times 2}$ and discuss their geometric behavior.

Ex 24 Linear Algebra

Towards SVD: Properties of A^TA and AA^T
Let $A \in \mathbb{R}^{m\times n}$ be any matrix. Please show:

1. $A^TA \in \mathbb{R}^{n\times n}$ and $AA^T \in \mathbb{R}^{m\times m}$ are symmetric.
2. A^TA and AA^T are positive semi-definite. (*Hint:* $\|x\|_2^2 = x^Tx \geq 0 \ \ \forall x \in \mathbb{R}^n$.)
   Remark: Thus, eigenvalues are nonnegative.
3. A^TA and AA^T have the same positive eigenvalues.
4. $\ker(A) = \ker(A^TA)$ and $\ker(A^T) = \ker(AA^T)$.
   Remark: Thus, $v \in \ker(A)$ is eigenvector of A^TA ($u \in \ker(A^T)$ is eigenvector of AA^T) to the eigenvalue $\lambda = 0$.
5. Name two sufficient conditions for the invertibility of A^TA and AA^T.

Ex 25

Let $A \in \mathbb{R}^{m \times n}$ be any matrix. Please show:

1. $A^T A$ is symmetric.
2. $A^T A$ is positive semi-definite.
3. $\ker(A) = \ker(A^T A)$.
   Hint: Show mutual subset relation.

Ex 26

Properties of $A^T A$ and AA^T
Let $A \in \mathbb{R}^{m \times n}$ be any matrix. Please show: Name two sufficient conditions for the invertibility of $A^T A$ and AA^T.

Ex 27 Linear Algebra

The Subspaces Kernel and Image

1. Let $A \in \mathbb{F}^{m \times n}$. Show that $\ker(A)$ and $\mathsf{Im}(A)$ are subspaces of $\mathbb{F}^n$ and $\mathbb{F}^m$, respectively.
2. Construct two example matrices and consider their kernel and image.

Ex 28 Linear Algebra

The Standard Inner Product
Show that the standard inner product $(\cdot,\cdot)\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, (x,y) \mapsto \sum_i^n x_i y_i$ satisfies

- (i) $\forall v, w \in \mathbb{R}^n : \ (v,w) = (w,v)$ (symmetric)
- (ii) $\forall v, w_1, w_2 \in \mathbb{R}^n : \ (v, w_1 + w_2) = (v,w_1) + (v,w_2)$
- $\forall v, w \in \mathbb{R}^n, r \in \mathbb{R} : \ (v, r \cdot w) = r \cdot (v,w)$ (linear in its second argument)
- (iii)$\forall v \in \mathbb{R}^n \backslash \{0\} : \ (v,v) > 0$ (positive definite)

Mappings $(\cdot,\cdot)\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ satisfying these three properties are called **inner products** or **symmetric bilinear forms** on $\mathbb{R}^n$.

Ex 29

Computing the Height of Objects
Let $a, b, c \in \mathbb{R}^2$ be vectors as illustrated in the figure below. We know from the lecture that the angle α between two vectors a, c is defined by

$$\cos(\alpha) = \frac{a^\top c}{\|a\|_2 \|c\|_2}.$$

Use this equality to compute the height $\|b\|_2$ of the tree in the figure below. The angle $\alpha := 36.87°$ and the distance to the tree $a := (4,0)^\top$ are given.

Ex 30 Linear Algebra

The Four Fundamental Subspaces

1. Let $A \in \mathbb{R}^{m \times n}$. Show that
   $$\forall y \in \mathsf{im}(A), x \in \ker(A^T)\colon y^T x = 0$$
   and
   $$\forall y \in \mathsf{im}(A^T), x \in \ker(A)\colon y^T x = 0.$$
2. Illuminate these findings on an example.

Remark: We say that $\ker(A^T)$ is the **orthogonal complement** of $\mathrm{im}(A)$ and write

$$\ker(A^T)^\perp = \mathrm{im}(A) \quad \text{or} \quad \ker(A^T) \perp \mathrm{im}(A) \quad \text{or} \quad \ker(A^T) = \mathrm{im}(A)^\perp.$$

Analogously $\ker(A)$ is the orthogonal complement of $\mathrm{im}(A^T)$ and we write

$$\ker(A)^\perp = \mathrm{im}(A^T) \quad \text{or} \quad \ker(A) \perp \mathrm{im}(A^T) \quad \text{or} \quad \ker(A) = \mathrm{im}(A^T)^\perp.$$

Ex 31 Linear Algebra

Linear Dependence

1. Give an example where a nontrivial linear combination of three nonzero vectors $a_1, a_2, a_3 \in \mathbb{R}^4$ is the zero vector (nontrivial means that not all scaling coefficients are zero).
2. Write your example in the form $Ax = 0$.

Ex 32 Linear Algebra

A matrix as a Linear Function
Let $A \in \mathbb{F}^{m\times n}$ be a matrix. Then consider the mapping $f_A \colon \mathbb{F}^n \to \mathbb{F}^m, x \mapsto Ax$.

1. Show that
$$f_A(\lambda x + y) = \lambda f_A(x) + f_A(y),$$
for all $x, y \in \mathbb{F}^n$ and $\lambda \in \mathbb{F}$.

   Hint: A vector is a matrix with just one column, so you can make use of the computation rules for matrices given in the lecture notes.

   Remark: Functions satisfying this property are called **linear**.
2. Use this fact to show the following equivalence:
$$\ker(A) := \{x \in \mathbb{F}^n \colon Ax = 0\} = \{0\} \quad \Leftrightarrow \quad f_A(x) = f_A(y) \text{ implies } x = y, \\ (\text{i.e., } f_A \text{ is an injective mapping}).$$

   Hint: Split up the equality $\Leftrightarrow$ into $\Rightarrow$ and $\Leftarrow$ and prove each of them separately.

Ex 33 Linear Algebra

Matrix-Norm
Let $A \in \mathbb{R}^{n\times m}$. Please show that $\|A\|_1 := \max\{\|Ax\|_1 : \|x\|_1 = 1\} = \max_j \sum_{i=1}^n |a_{ij}|$.

Hint: As usual it is helpful to split up the equality sign into $\leq$ and $\geq$ and treat the parts separately.

Ex 34 Linear Algebra

Matrix Product as Sum of rank-1 Matrices

1. Let $A \in \mathbb{R}^{m\times k}$ and $B \in \mathbb{R}^{k\times n}$. Show that
$$A \cdot B = \sum_{i=1}^k a_i b_i^\top = \sum_{i=1}^k \begin{pmatrix} a_{1i} \\ \vdots \\ a_{mi} \end{pmatrix} \cdot \begin{pmatrix} b_{i1} & \dots & b_{in} \end{pmatrix},$$
   where $a_i \in \mathbb{R}^{m\times 1}$ denotes the i-th *column* of A and $b_i^\top \in \mathbb{R}^{1\times n}$ denotes the i-th *row* of B.
2. Construct two examples with actual numbers.

Remark: Also see p. 10-11 in Gilbert Strang's "Linear Algebra and Learning from Data".

Ex 35 Linear Algebra

Computation Rules for Matrices and Vectors

Below you find a collection of computation rules that are helpful when dealing with matrices and vectors. Feel free to prove them based on the definitions given in the lecture.

Compatibility properties of summing and scaling matrices
Let $A, B \in \mathbb{F}^{m\times n}$ and $r, s \in \mathbb{F}$. Then

$$
\begin{aligned}
&i) && (r \cdot s) \cdot A = r \cdot (s \cdot A)\\
&ii) && (r + s) \cdot A = r \cdot A + s \cdot A\\
& && r \cdot (A + B) = r \cdot A + r \cdot B\\
&iii) && 1 \cdot A = A
\end{aligned}
$$

From which we can derive:

$$
\begin{aligned}
&i) && 0 \cdot A = 0\\
&ii) && r \cdot 0 = 0\\
&iii) && r \cdot A = 0 \quad \Rightarrow r = 0 \ \vee \ A = 0\\
&iv) && (-1) \cdot A = -A
\end{aligned}
$$

Compatibility properties of matrix sum and product
Let $A, \tilde{A} \in \mathbb{F}^{m\times n}$, $B, \tilde{B} \in \mathbb{F}^{n\times l}$, $C \in \mathbb{F}^{l\times t}$, $r \in \mathbb{F}$. Then

$$
\begin{aligned}
&i)\ (A \cdot B) \cdot C = A \cdot (B \cdot C)\\
&ii)\ (A + \tilde{A})B = AB + \tilde{A}B\\
&iii)\ A(B + \tilde{B}) = AB + A\tilde{B}\\
&iv)\ I_m A = AI_n = A\\
&v)\ (r \cdot A) \cdot B = r(A \cdot B) = A(r \cdot B)\\
&vi)\ 0A = A0 = 0
\end{aligned}
$$

Group property of invertible matrices
For two invertible matrices $A, B \in \mathbb{F}^{n\times n}$ we find

$$
\begin{aligned}
&i)\ (AB)^{-1} = B^{-1}A^{-1}\\
&ii)\ \left(A^{-1}\right)^{-1} = A
\end{aligned}
$$

Transpose matrices
Let $A \in \mathbb{R}^{m\times r}$ and $B \in \mathbb{R}^{r\times m}$. Then

i) $(A^\top)^\top = A$,

ii) $(AB)^\top = B^\top A^\top$,

iii) $(A + B)^\top = A^\top + B^\top$,

iv) for $A \in GL(n, \mathbb{R})$ we have $(A^\top)^{-1} = (A^{-1})^\top$.

Ex 36 Linear Algebra

Linear Modeling
Let us assume a car c_1 starts driving at time $t = 0h$ and runs with $50\frac{km}{h}$. A second car c_2 runs with $90\frac{km}{h}$ but starts 2 hours later at time $t = 2h$ at the same position as car c_1 did.

1. Describe the positions $c_1(t)$ and $c_2(t)$ of the cars as affine linear function of time, i.e., $c_i(t) = m_i t + b_i$ for some appropriate $m_i, b_i \in \mathbb{R}$, $i = 1, 2$.
2. At which time t do the cars meet?

Ex 37

Let $A \in \mathbb{R}^{n\times n}$ be a *negative* definite matrix, i.e., $x^\top Ax < 0$ for all $x \in \mathbb{R}^n \setminus \{0\}$. Show that the eigenvalues of A are strictly negative, i.e., $\lambda < 0$ for all $\lambda \in \sigma(A)$.

Ex 38 Linear Algebra

$(n + 1)$ **vectors are always dependent**

Consider arbitrary $n+1$ vectors $v_1, \dots, v_{n+1} \in \mathbb{F}^n$. Then there are coefficients $\alpha_i \in \mathbb{F}, i = 1, \dots, n+1$, which are not all zero and solve the equation $\sum_{i=1}^{n+1} \alpha_i v_i = 0$.

Ex 39 Linear Algebra

Orthogonal Matrices
A matrix $Q \in \mathbb{R}^{n \times n}$ is called **isometric**, if

$$\|Qx\|_2 = \|x\|_2, \quad \forall x \in \mathbb{R}^n.$$

1. **Bonus** (You can get 4 bonus points for this subtask): Show the following equivalence:

   $$Q \text{ is isometric} \quad \Leftrightarrow \quad Q \text{ is orthogonal.}$$

   Hint: To show "⇒", use the **polarization identity**

   $$(x, y)_2 = \frac{1}{4} \left(\|x + y\|^2 - \|x - y\|^2 \right), \quad x, y \in \mathbb{R}^n$$

   and the observation

   $$a_{ij} = (e_i, Ae_j)_2, \quad \text{for} \quad A = (a_{ij})_{ij} \in \mathbb{R}^{n \times n},$$

   where $e_j = (0, \dots, 1, \dots, 0)^T \in \mathbb{R}^n$ denote the standard basis vectors.

2. For an angle $\alpha \in [0, 2\pi]$, consider the *rotation matrix*

   $$Q_\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}.$$

   a) Show that Q_α is orthogonal for all α.
   Hint: Use the Pythagorian identity $\sin^2\alpha + \cos^2\alpha = 1$

   b) Draw the set $\{Q_\alpha x : \|x\|_1 = 1\}$ for $\alpha = \pi/4$.

Ex 40 Linear Algebra

The p-Norm
A mapping $\|\cdot\| : \mathbb{R}^n \to [0, +\infty)$ is called **norm** on $\mathbb{R}^n$ if it satisfies the three properties

i) $\|x\| = 0 \Rightarrow x = 0$ *(positive definite/ point separating)*

ii) $\|r \cdot x\| = |r| \cdot \|x\|, \ \forall x \in \mathbb{R}^n, r \in \mathbb{R}$ *(absolutely homogeneous)*

iii) $\|x + y\| \leq \|x\| + \|y\|, \ \forall x, y \in \mathbb{R}^n$ *(subadditive/ triangle inequality)*

The most common example is given by the p-**norm** for $p \in [1, +\infty)$, defined by

$$\|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

and the **supremum (or maximum) norm** defined by

$$\|x\|_\infty := \max_{i=1,\dots,n} |x_i|.$$

Remark: For a fixed $x \in \mathbb{R}^n$ one can show $\lim_{p \to \infty} \|x\|_p = \|x\|_\infty$.

Tasks:

1. Draw the sets $\{x \in \mathbb{R}^2 : \|x\|_p = 1\}$ for $p = 1, 2, \infty$.

2. Show that the Euclidean norm $\|\cdot\|_2 : \mathbb{R}^n \to [0, +\infty), \ x \mapsto \sqrt{\sum_{i=1}^n x_i^2} = \sqrt{x^\top x}$ satisfies i)-iii).
   Hint: For iii) use the Cauchy-Schwarz inequality from the lecture and show $\|x + y\|^2 \leq (\|y\|_2 + \|x\|_2)^2$ and also use $|a + b| \leq |a| + |b|$ for $a, b \in \mathbb{R}$ (i.e., the triangle inequality is true on $\mathbb{R}^1$).

Ex 41 SciPy Stack, Python

NumPy and Matplotlib II

In this exercise you need to make use of the Python packages numpy and `matplotlib`.

1. Construct an identity matrix `I` of dimension 100x100 as `numpy.ndarray`.
2. Construct a banded matrix `A` of the form

$$A = \frac{100^2}{4\pi^2} \begin{bmatrix} -2 & 1 & 0 & 0 & \dots & 0 & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 & 0 \\ 0 & 0 & 1 & -2 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & -2 & 1 \\ 0 & 0 & 0 & 0 & \dots & 1 & -2 \end{bmatrix}$$

   of dimension 100x100 as `numpy.ndarray`.
3. Plot both matrices with the function `imshow()` of the package `matplotlib.pyplot`.
4. Construct a vector z with the `linspace()` function of the numpy package. It should contain a grid of 102 values between 0 and 2π.
5. Use Python's *slicing* capabilities to copy all except from the first and the last value of z into another vector x.
6. Calculate $y = \sin(x)$ and the matrix vector product $d = Ay$ using `numpy.sin()` and `numpy.dot()`.
7. Plot y and d into the same plot using `plot()` from the `matplotlib.pyplot` package.

Hint: You might need to use `matplotlib.pyplot.show()` in order to guarantee that the notebook shows some output.

Ex 42 Linear Algebra, Python

The Matrix-Vector Product

Implement a function that takes as input a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $x \in \mathbb{R}^n$ and then returns the matrix-vector product Ax.

1. Implement the following four ways:
   a) **Dense:** Input expected as `numpy.ndarray`:
   Assume that the matrix and the vector are delivered to your function as `numpy.ndarray`.
      i. Implement the matrix-vector product "by hand" using for loops, i.e., *without* using numpy functions/methods.
      ii. Implement the matrix-vector product using `A.dot(x)`, `A@x`, `numpy.matmul(A,x)` or `numpy.dot(A,x)`.
   b) **Sparse:** Matrix expected in CSR format:
   Assume that the matrix is delivered to your function as `scipy.sparse.csr_matrix` object. The vector x can either be expected as `numpy.ndarray` or simply as a Python `list`.
      i. Access the three CSR lists via `A.data`, `A.indptr`, `A.indices` and implement the matrix-vector product "by hand" using for loops.
      ii. Implement the matrix-vector product using `A.dot(x)` or `A@x` .
2. **Test** your four different routines from above on the following matrix $A \in \mathbb{R}^{n \times n}$ with constant diagonals given by

$$A = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

   and the input vector

$$x = (1, \cdots, 1)^\top \in \mathbb{R}^n \quad \text{(you can use: } \texttt{x = numpy.ones(n)}\text{)}.$$

   a) Determine how $b := A \cdot x \in \mathbb{R}^n$ looks like in this example in order to facilitate a test.

b) Test whether your four routines compute the matrix–vector product correctly by checking $A \cdot x = b$.

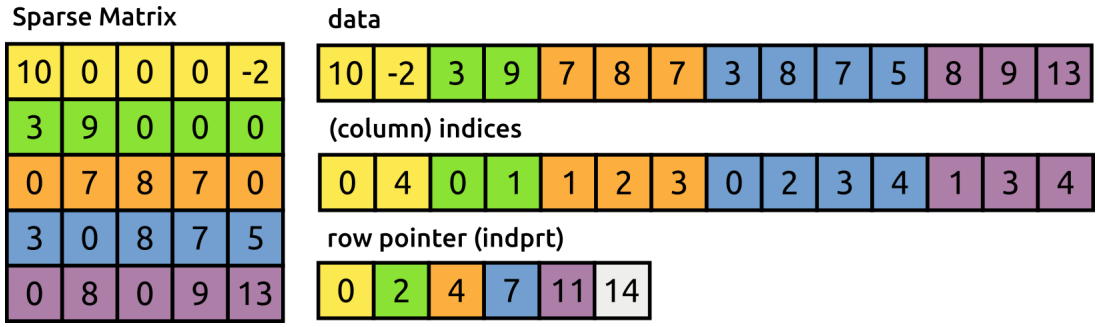
c) Use different values for the dimension n (especially large $n \geq 10^5$ – note that you may exceed your hardware capacities for the dense computations).

Remark: The matrix has "2" on the main diagonal and "-1" on the first off-diagonals.

3. For all cases:

a) **Memory:** What is the number of Gbytes needed to store an $m \times n$ array of `floats`? Print the number of Gbytes which are needed to store the matrix in all cases.
Hint: A number implemented as `float` in Python implements double precision and therefore needs 64 Bits of storage. For a `numpy.ndarray` you can type `A.nbytes` and for the `scipy.sparse.csr_matrix` you can type `A.data.nbytes + A.indptr.nbytes + A.indices.nbytes`.

b) **Computation times:** Measure the time which is needed in each case to compute the matrix-vector product for a random input vector `x = numpy.random.rand(n)`.
Hint: In the IPython shell you can simply use the *magic function* `%timeit` to measure the time for a certain operation. For example, you can type `%timeit pythonfunction(x)`. Alternatively you can use the package `timeit`.



Example of a Matrix in CSR Format

Ex 43 Linear Algebra

Property of a skew-symmetric Matrix
A matrix $C \in \mathbb{R}^{n \times n}$ is called *skew-symmetric* if

$$C^\top = -C.$$

Please show that for a real-valued skew-symmetric matrix $C \in \mathbb{R}^{n \times n}$ it holds

$$x^\top C x = 0$$

for all $x \in \mathbb{R}^n$ (i.e., a vector x is always mapped to a perpendicular vector Cx).

Ex 44 Linear Algebra

Rank/Image and Nullity/Kernel
Consider the matrix

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix},$$

the column vector $\mathbf{1} := \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ (i.e., a (3×1) matrix) and the row vector $\tilde{\mathbf{1}} := (1\ 1\ 1)$ (i.e., a (1×3) matrix).

1. Show that $A = \mathbf{1}\tilde{\mathbf{1}}$.
2. Find two distinct nonzero vectors x and y, so that $Ax = 0$ and $Ay = 0$.
3. How does the image $\mathrm{Im}(A)$ look like? First draw a picture. Then find a basis of $\mathrm{Im}(A)$ to determine the rank of the matrix.
4. How does the kernel $\ker(A)$ look like? First draw a picture. Then find a basis of $\ker(A)$ to determine the nullity of the matrix.

   Remark: You have to prove that your vectors are a basis.

Ex 45 Linear Algebra

The Inner Product and SPD Matrices

- A mapping $(\cdot,\cdot)\colon \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}$ is called **inner product** on $\mathbb{F}^n$ if it satisfies:
  i) Hermitian: $\forall x, y \in \mathbb{F}^n :\ (x,y) = \overline{(y,x)}$,
  ii) Linear in its second argument: $\forall x, y_1, y_2 \in \mathbb{F}^n, \lambda \in \mathbb{F} :\ (x, y_1 + \lambda y_2) = (x, y_1) + \lambda(x, y_2)$,
  iii) Positive definite: $\forall x \in \mathbb{F}^n \backslash \{0\} :\ (x,x) > 0$.
- A hermitian matrix $A \in \mathbb{F}^{n\times n}$ $(A^H = A)$ is called **positive definite** (p.d.), if

$$x^\top A x > 0 \quad \forall x \in \mathbb{F}^n \setminus \{0\}.$$

- We find the following relation:

$$(\cdot,\cdot)\colon \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F} \text{ is inner product } \iff \exists A \in \mathbb{F}^{n\times n} \text{ hermitian and p.d.}\colon (x,y) = x^T A y.$$

  In the lecture so far we considered the *standard inner product*

$$(\cdot,\cdot)_2\colon \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F},\ (x,y) \mapsto x^H I_n y = \sum_{i=1}^n \overline{x_i} y_i$$

  which corresponds to the hermitian and positive definite matrix I_n (identity matrix).

Tasks:

1. Consider $\mathbb{F} = \mathbb{R}$. Find an example $A \in \mathbb{R}^{2\times 2}$ which is symmetric and positive definite.
2. Draw the set $\{Ax\colon \|x\|_2 = 1\}$.

Ex 46 Linear Algebra

SPD Matrices

Definition. Let $A \in \mathbb{R}^{n\times n}$ be a matrix.

- A is called **symmetric**, if $A^\top = A$.
- A is called **positive definite (semi-definite)**, if $x^\top A x > 0$ $(x^\top A x \geq 0)$ for all $x \neq 0$.

Remark: If A is symmetric and positive definite (SPD), then $(x,y) := x^T A y$ defines a general so-called *inner product*. In the lecture we mainly consider the *standard inner product*

$$(\cdot,\cdot)_2\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R},\ (x,y) \mapsto x^T I_n y = \sum_{i=1}^n x_i y_i$$

which corresponds to the symmetric and positive definite matrix I_n (identity matrix).

Tasks:

1. Find an example $A \in \mathbb{R}^{2\times 2}$ which is symmetric and positive definite.
2. Draw the set $\{Ax\colon \|x\|_2 = 1\}$.

Ex 47 Linear Algebra

Thin and Fat Full Rank Matrices

Answer the following questions without using the SVD. Instead, exploit the orthogonality relation between the four fundamental subspaces and the dimension formula.

1. What is the orthogonal complement of $\{0\}$ and $\mathbb{R}^n$ in $\mathbb{R}^n$, respectively?
2. Give an example for a matrix $C \in \mathbb{R}^{m\times n}$ with nullity$(C) = 0$ (injective).

a) What do we know about the order relation between m and n? (which one is greater or equal than the other?)

b) What do we know about the columns of C?

c) Let $b \in \text{im}(C)$. Can we find two distinct $x_1 \neq x_2 \in \mathbb{R}^n$ such that $Cx_1 = b = Cx_2$? Explain your answer.

d) What do we know about the matrix $C^\top C \in \mathbb{R}^{n \times n}$?

3. Give an example for a matrix $A \in \mathbb{R}^{m \times n}$ with $n > m$ and $\text{rank}(A) < m$.

4. Give an example for a matrix $R \in \mathbb{R}^{m \times n}$ with $\text{rank}(R) = m$ (surjective).

a) What do we know about the order relation between m and n?

b) What do we know about the rows of R?

c) What do we know about the matrix $RR^\top \in \mathbb{R}^{m \times m}$?

d) Let $b \in \mathbb{R}^m$. Can we find an $x \in \mathbb{R}^n$ such that $Rx = b$? Explain your answer and give an example for your matrix.

5. Give an an example for a matrix $A \in \mathbb{R}^{n \times n}$ with $\text{rank}(A) = n$ (invertible).

a) What do we know about the dimensions of $\ker(A)$, $\ker(A^\top)$ and $\text{im}(A^\top)$?

b) What do we know about the columns and rows of A?

c) Let $b \in \mathbb{R}^n$. Can we find an *unique* $x \in \mathbb{R}^n$ such that $Ax = b$? Explain your answer and give an example for your matrix.

6. **Bonus*:** Give an an example for a matrix $A \in \mathbb{R}^{m \times n}$ with $\text{rank}(A) = \text{rank}(A^\top)$ and $\text{nullity}(A) \neq \text{nullity}(A^\top)$.

Ex 48 Linear Algebra

Computation Rules for Matrices II
Let $A \in \mathbb{R}^{m \times r}$ and $B \in \mathbb{R}^{r \times m}$. Then

i) $(A^\top)^\top = A$,

ii) $(AB)^\top = B^\top A^\top$,

iii) $(A + B)^\top = A^\top + B^\top$,

iv) for $A \in GL(n, \mathbb{R})$ we have $(A^\top)^{-1} = (A^{-1})^\top$.

Ex 49 Linear Algebra

Height of a tree
Let $a, b, c \in \mathbb{R}^2$ be vectors as illustrated in the figure below. We know from the lecture that the angle α between two vectors a, c is defined by

$$\cos(\alpha) = \frac{a^\top c}{\|a\|_2 \|c\|_2}.$$

Use this equality to compute the height $\|b\|_2$ of the tree in the figure below. The angle $\alpha := 36.87°$ and the distance to the tree $a := (4, 0)^\top$ are given.

---

3 Solving Linear Systems

Ex 50

Curve Fitting
Assume you were given the following data:

z	-1	0	1
y	-1	3	1

1. **Polynomial Interpolation:**
   Find coefficients $c_i \in \mathbb{R}$ for the following parabola (polynomial of degree 2)
   $$f(z) = c_0 + c_1 z + c_2 z^2,$$
   such that $f(z_i) = y_i$ for all three measurements $i = 1, 2, 3$ from the table above. Therefore, set up and solve a linear system of the form $Ax = b$, where $x = (c_0, c_1, c_2)$. Draw the measurements and the parabola into one plot.

2. **Univariate Linear Regression:**
   Now assume a linear model of the form
   $$f(z) = c_0 + c_1 z.$$
   Consider $x = (c_0, c_1)$. Determine a matrix A and a vector b, such that
   $$\sum_{i=1}^{3} (f(z_i) - y_i)^2 = \|Ax - b\|_2^2.$$
   Solve the least squares problem
   $$\min_{x \in \mathbb{R}^2} \|Ax - b\|_2^2,$$
   to determine the coefficients $x = (c_0, c_1)$ by solving the *normal equation* (explanation follows later in the lecture)
   $$A^T A x = A^T b.$$
   Draw the measurements and the straight line into one plot.

Ex 51 Linear Algebra

Gram-Schmidt-Algorithm
The Gram-Schmidt algorithm is an algorithm to compute a QR-decomposition of some matrix A. The basic idea is to successively built up an orthogonal system from a given set of linearly independent vectors. Those are, in this case, given by the columns of an invertible matrix $A = [a_1, \ldots, a_n] \in \mathbb{R}^{n \times n}$. We choose the first column as starting point for the algorithm and set $\widetilde{q_1} := a_1$. Of course, in order to generate an orthogonal matrix Q we have to rescale the vector and set $q_1 := \frac{\widetilde{q_1}}{\|\widetilde{q_1}\|}$. The successive vectors $\widetilde{q_k}$ are generated by subtracting all the shares $a_k^\top q_\ell$ of the previous vectors q_ℓ from the column a_k, i.e.
$$\widetilde{q_k} := a_k - \sum_{\ell=1}^{k-1} a_k^\top q_\ell \; q_\ell.$$

The following algorithm computes a QR-decomposition of some matrix $A \in \mathbb{R}^{n \times n}$.

1. Please check that the matrix $Q := [q_1, \ldots, q_n] \in \mathbb{R}^{n \times n}$ is orthogonal, i.e. that $Q^T Q = I_n$.
   Hint: Show $\|q_i\| = 1$ first, and then perform an induction proof using the induction assumption that q_k is orthogonal to all previous $q_1, \ldots, q_{k-1}$.
2. Let $R := (r_{\ell k})_{\ell \leq k}$ be the upper triangular matrix which results from the Gram-Schmidt algorithm. Please show that the algorithm provides a QR decomposition, i.e., that $QR = A$.
   Hint: It suffices to show $Qr_k = a_k$, where r_k (a_k) is the k-th column of R (A).

$r_{11} \leftarrow \|a_1\|$;
$q_1 \leftarrow \frac{a_1}{r_{11}}$;
for $k = 2, \ldots, n$ **do**
  for $\ell = 1, \ldots, k-1$ **do**
    $r_{\ell k} \leftarrow a_k^\top q_\ell$
  end
  $\widetilde{q}_k \leftarrow a_k - \sum_{\ell=1}^{k-1} r_{\ell k}\, q_\ell$;
  $r_{kk} \leftarrow \|\widetilde{q}_k\|$;
  $q_k \leftarrow \frac{\widetilde{q}_k}{r_{kk}}$;
end

Algorithm 1: Gram-Schmidt algorithm

Ex 52

****Bonus:** Operation Count in Classical Gram Schmidt Algorithm**
Consider the classical Gram-Schmidt algorithm for a matrix $A \in \mathbb{R}^{m\times n}$ with $\ker(A) = \{0\}$.
Count each addition, subtraction, multiplication, division and square root that is computed in the classical Gram Schmidt algorithm to obtain a *reduced* QR factorization of A.

Ex 53

Consider the system

$$\begin{pmatrix} \frac{1}{2} & -2 & 0 \\ 2 & 8 & -2 \\ 1 & 0 & 2 \end{pmatrix} x = \begin{pmatrix} -1 \\ 10 \\ 4 \end{pmatrix}.$$

1. Define a matrix A and a vector b, so that this system reads as $Ax = b$. Then compute an LU-decomposition of A by applying Gaussian elimination with **row pivoting**. Denote the respective matrices L, U and P, such that $PA = LU$. (*Hint:* Verify the desired properties of the factor matrices and test whether $PA = LU$ holds.)
2. Use the result from the LU-decomposition to determine an x which solves $Ax = b$.
   (*Hint:* Test whether $Ax = b$ holds.)

Ex 54

Consider the system

$$\begin{pmatrix} 0 & -2 & 1 \\ 2 & 2 & 0 \\ -2 & -\frac{3}{2} & 0 \end{pmatrix} x = \begin{pmatrix} -5 \\ 6 \\ -5 \end{pmatrix}.$$

1. Define a matrix A and a vector b, so that this system reads as $Ax = b$. Then compute an LU-decomposition of A by applying Gaussian elimination with **row pivoting**. Denote the respective matrices L, U and P, such that $PA = LU$. (*Hint: Verify the desired properties of the factor matrices and test whether $PA = LU$ holds.*)
2. Use the result from the LU-decomposition to determine an x which solves $Ax = b$.
   (*Hint: Test whether $Ax = b$ holds.*)

Ex 55

Solving Linear Systems using LU Decomposition
Consider the following linear systems.

1.

$$\begin{aligned} 2x_1 + x_2 + 3x_3 &= -3 \\ x_1 - x_2 - x_3 &= 4 \\ 3x_1 - 2x_2 + 2x_3 &= 5 \end{aligned}$$

2.

$$\begin{aligned} x_1 + 2x_2 + 2x_3 &= 1 \\ 2x_1 + x_3 &= 3 \\ 3x_1 + 2x_2 + 3x_3 &= 4 \end{aligned}$$

3.

$$\begin{aligned} x_1 + x_2 + 2x_3 &= 2 \\ 1x_1 - x_2 &= 0 \\ 2x_1 + 2x_3 &= 1 \end{aligned}$$

Cast them into the form $Ax = b$ and compute the LU-decomposition of A by rigorously applying Algorithm 3 (i.e., use the pivot element determined in Line 8):

- Find the matrices L, U and P such that $PA = LU$.
- Then determine the solution set $S := \{x \in \mathbb{R}^n : Ax = b\}$.

Hint: System 2. does not have a *unique* solution (but infinitely many). Determine the set of vectors $x \in \mathbb{R}^3$ for which the linear system 2. is valid.

1 **INPUT:** $A \in \mathbb{R}^{n \times n}$ (and $b \in \mathbb{R}^n$)
2 **OUTPUT:** LU decomposition $PA = LU$ (and if $Ax = b$ is uniquely solvable the solution $x \in \mathbb{R}^n$)
3
4 `# FACTORIZATION`
5 initialize `piv = [1,2,...,n]`
6 **for** $j = 1, ..., n-1$ **do**
7 `# Find the j-th pivot`
8 $k_j := \arg\max_{k \geq j} |a_{kj}|$
9 **if** $a_{k_j j} \neq 0$ **then**
10 `# Swap rows`
11 A[k_j,:] $\leftrightarrow$ A[j,:]
12 `# by hand we also transform b on the fly`
13 b[k_j] $\leftrightarrow$ b[j]
14 piv[k_j] $\leftrightarrow$ piv[j]
15 `# Elimination`
16 **for** $k = j+1, \ldots, n$ **do**
17 $\ell_{kj} := a_{kj}/a_{jj}$
18 $a_{kj} = \ell_{kj}$
19 **for** $i = j+1, \ldots, n$ **do**
20 $a_{ki} = a_{ki} - \ell_{kj} a_{ji}$
21 **end**
22 `# by hand we also transform b on the fly`
23 $(b_k = b_k - \ell_{kj} b_j)$
24 **end**
25 **end**
26 **end**
27
28 `# SOLVE`
29 ...

Algorithm 2: In-place Gaussian Elimination with Row Pivoting: Factor and Solve

---

Ex 56

Proof for LU decomposition (with row pivoting)
Let $m \in \mathbb{N}$. As above, let $P_{ik_i} \in \mathbb{R}^{m \times m}$ be the permutation matrix which results from interchanging the i-th and k_i-th column ($k_i \geq i$) of the identity matrix in $\mathbb{R}^{m \times m}$. Further for $\ell_j := (0, \ldots, 0, \ell_{j+1,j}, \ldots, \ell_{m,j})^\top \in \mathbb{R}^m$ and the j-th unit vector $e_j \in \mathbb{R}^m$, let $L_j := I + \ell_j e_j^\top \in \mathbb{R}^{m \times m}$. Then show that for all $1 \leq j < i \leq k_i \leq m$ we have

$$P_{ik_i} L_j = \widehat{L}_j P_{ik_i}$$

where $\widehat{L}_j := I + (P_{ik_i}\ell_j)e_j^\top$.

Ex 57 Linear Systems, Factor and Solve

Permutation Matrices and Row Swap
A matrix $P \in \mathbb{R}^{m\times m}$ is called *permutation matrix* if it has exactly one entry of 1 in each row and each column and 0s elsewhere.

1. **Sparse representation (CSR Format):** A permutation matrix $P = (p_{ij})_{ij} \in \mathbb{R}^{m\times m}$ can be represented by an m-dimensional vector piv$\in \{1, 2, \ldots, m\}^m$ in the following way:

   $$\texttt{piv}_i = j \quad :\Leftrightarrow \quad p_{ij} = 1.$$

   Given a permutation matrix P with its sparse representation piv. Determine the sparse representation, say pivT$\in \{1, 2, \ldots, m\}^m$, of the transpose $P^T = (\widetilde{p}_{ij})_{ij}$, so that

   $$\texttt{pivT}_i = j \quad :\Leftrightarrow \quad \widetilde{p}_{ij} = 1.$$

   Hint: Have a look at the routine `numpy.argsort()`.

2. **Inverse:** Show that the inverse of a permutation matrix $P \in \mathbb{R}^{m\times m}$ is given by its transpose, i.e., $P^T = P^{-1}$.

3. **Row Swap**: Let $P_{jk} \in \mathbb{R}^{m\times m}$ be the permutation matrix which results from interchanging the j-th and k-th column ($k \geq j$) of the identity matrix in $\mathbb{R}^{m\times m}$. Thus if its applied to a matrix $A \in \mathbb{R}^{m\times n}$ it interchanges the j-th and k-th row of A. Show that

   $$P_{jk}^\top = P_{jk}.$$

   In particular we find $P_{jk} = P_{jk}^{-1}$, i.e., P_{jk} is self-inverse.

Ex 58 Linear Systems, Factor and Solve, Python

The Cholesky Decomposition for SPD Matrices

1. Find SciPy routines to perform the Cholesky factorization and solution steps separately. Non-obligatory: Implement them on your own (algorithms can be found on Wikipedia).

2. Compare the performance of the Cholesky routines to the LU routines from SciPy by testing them on large symmetric and positive definite matrices $A \in \mathbb{R}^{n\times n}$ (e.g., $A = B^TB + \delta I$ for some random $B \in \mathbb{R}^{n\times n}$ and $\delta > 0$) and some (random) right-hand side $b \in \mathbb{R}^n$.

Ex 59

Curve Fitting using reduced QR Factorization

Assume you were given some `numpy.ndarray` called "data" containing measurements $(z_1, y_1), \ldots, (z_m, y_m) \in \mathbb{R} \times \mathbb{R}$. Further assume that this data has shape `(2,m)` so that the first row `data[0,:]` contains the z_i values and the second row `data[1,:]` contains the y_i values.

1. Implement a function `poly_curvefit(data, p)` which computes a polynomial fit to the data.

   The **input** data shall have the form as described above and p shall determine the polynomial used to fit the data. More precisely, p shall be a list `[p1, ..., pn]` containing $n \leq m$ *distinct* natural numbers between 0 and $m-1$ which determine the polynomial model f by

   $$f(z) = c_1 z^{p_1} + c_2 z^{p_2} + \ldots + c_k z^{p_n} = \sum_{j=1}^{n} c_j f_j(z) \quad \text{with} \quad f_j(z) := z^{p_j}.$$

   For example, `p = [0,1]` amounts for a linear model

   $$f(z) = c_1 z^{p_1} + c_2 z^{p_2} = c_1 z^0 + c_2 z^1 = c_1 + c_2 z^1.$$

   Then the **function** `poly_curvefit(data, p)` shall determine appropriate coefficients $x = (c_1, \ldots, c_n)$ by following this recipe:

a) Assemble the vector $b = (y_1, \dots, y_m) \in \mathbb{R}^m$.

b) Assemble the matrix $A = (a_{ij})_{ij} \in \mathbb{R}^{m \times n}$, where $a_{ij} := f_j(z_i)$.

c) The vector b may not be in the image of A. Therefore compute the reduced QR factorization of A and solve the auxiliary problem $\widehat{R}x = \widehat{Q}^T b$. You can use `qr_factor` and `solve_tri` from previous exercises (or appropriate SciPy routines).
Remark: We will later learn that this yields precisely the least squares solution.

2. Test your routine by fitting the data

z	-2	-1	0	1	2
y	-2	1	-1	2	6

with different polynomials of your choice. Plot the measurements and the fitting polynomial into one figure.

Ex 60 Linear Algebra, Python

Classical Gram Schmidt Algorithm

Let $A \in \mathbb{R}^{m \times n}$ with $\ker(A) = \{0\}$. Then the classical Gram Schmidt algorithm to compute a *reduced* QR-decomposition of A reads as follows:

$r_{11} = \|a_1\|$
$q_1 = \frac{a_1}{r_{11}}$
for $k = 2, \dots, n$ **do**
  for $\ell = 1, \dots, k-1$ **do**
    $r_{\ell k} = a_k^\top q_\ell$
  end
  $\widetilde{q}_k = a_k - \sum_{\ell=1}^{k-1} r_{\ell k}\, q_\ell$
  $r_{kk} = \|\widetilde{q}_k\|$
  $q_k = \frac{\widetilde{q}_k}{r_{kk}}$
end

Task:

1. Implement this algorithm as a function `Q, R = qr_factor(A)`, which takes a matrix A as input and returns the matrices $\widehat{Q}$ and $\widehat{R}$.

2. Legendre polynomials: Run your algorithm on the following example.
   a) Find a NumPy function to generate an equidistant grid on the interval $(-1, 1)$.
   b) Find a NumPy function to generate a Vandermonde matrix `A` based on this grid.
   c) Compute `Q,R = qr_factor(A)` and plot the columns of `A` and `Q` over the interval $(-1, 1)$ into one figure.
   d) Compute `Q.TQ` and `Q@R - A` and check whether the first yields the identity and the latter the zero-matrix.
   Hint: You can use `numpy.allclose()` *to check whether two* `numpy.ndarray`*'s are equal up to a certain tolerance.*

3. Find a SciPy function to compute a full QR factorization. To obtain the reduced QR you need to set the parameters accordingly.

Ex 61 Linear Systems, Python

Implementation Matters
In order to solve the problem $Ax = b$, there are plenty of algorithms available. In this exercise we invoke several SciPy routines to solve linear systems numerically. Thereby, we will learn that different algorithms, or even different implementations of the same algorithm, can have a huge effect on the efficiency.

Consider the following test example: The matrix $A \in \mathbb{R}^{n \times n}$ with constant diagonals given by

$$A = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

and the right-hand side vector

$$b = (1, 0, \cdots, 0, 1)^\top \in \mathbb{R}^n.$$

In this case we know that the unique[1] solution $x^* = A^{-1}b$ is given by

$$x^* = (1, 1, \cdots, 1)^\top \in \mathbb{R}^n.$$

Implement the following options to solve the problem $Ax = b$ (i.e., given A and b from above, find a numerical solution $\tilde{x}$ such that $A\tilde{x} \approx b$):

1. Dense: Work with A as dense `numpy.ndarray`.
   a) The forbidden way: Find a SciPy function to compute a numerical inverse and apply it to b to obtain $\tilde{x}$.
   b) The default way: Find a general solver for linear systems.
   c) Structure exploiting solver I: Find a way to inform this general solver about the fact that A is *positive definite*.[2]
   d) Structure exploiting solver II: Find another solver which exploits the fact, that A has constant diagonals.
2. Sparse: Exploit the sparsity of the matrix and work with A as `scipy.sparse.csr_matrix`.
   a) The forbidden way: Find a SciPy function to compute the inverse of a sparse matrix and apply it to b to obtain $\tilde{x}$.
   b) The default way: Find a general solver for sparse linear systems.
   c) Structure exploiting solver: Find a function that implements the *conjugate gradient (cg)* method to solve the system iteratively. Play with the optional parameter `<maxiter>` to restrict the number of iterations. What do you observe?

Run your code for different dimensions n (especially large $n \geq 10^5$) and:

1. measure the time needed for each of your solving routine,
2. and find a SciPy function to compute the error $\frac{1}{n}\|\tilde{x} - x^*\|_2$ for each solving routine.

Remark: For now, it is not important to understand how the algorithms work. We'll learn more about them throughout this course. Here, you learn to find appropriate SciPy functions which solve your problem and to serve the interfaces of these functions.

Ex 62 Linear Systems, Factor and Solve, Python

Solve with LU decomposition [Direct method]

1. Implement a routine `lu,piv = lu_factor(A)` which computes the LU decomposition $PA = LU$ (by applying Gaussian elimination with row pivoting; see Algorithm 3) for a given matrix $A \in \mathbb{R}^{n \times n}$ and another routine `lu_solve((lu, piv), b)` which takes the output of `lu_factor(A)` and returns the solution x of $Ax = b$ for some $b \in \mathbb{R}^n$ (in case the system admits an *unique* solution).
   - Store L and U in one array `lu` and the permutation P as sparse representation in an array `piv`.
   - If the system $Ax = b$ admits an unique solution then compute it by using your routine `solve_tri` from previous exercises or an appropriate SciPy routine. If the system is not uniquely solvable, check whether the system has infinitely many or no solution and give the user a respective note.
   - *Hint:* With the numpy routines `numpy.triu` and `numpy.tril` you can extract the factors L and U from the array `lu`. Also observe that we expect A to be of square format (for simplicity).
2. Test your routine at least on the systems which you were asked to solve by hand previously. Verify that $PA = LU$ and potentially $Ax = b$. For this purpose you can use `numpy.allclose()`.
3. Find SciPy routines to perform the factorization and solution steps and compare to your routine.

Ex 63 Linear Systems, Factor and Solve, Python

Solving Linear Systems with Triangular Matrices

[1] One can show that the matrix A is invertible.
[2] We'll learn about this property later.

INPUT: $A \in \mathbb{R}^{n \times n}$
OUTPUT: LU decomposition $PA = LU$

```
# FACTORIZATION
initialize piv = [1,2,...,n]
for j = 1,...,n-1 do
    # Find the j-th pivot pivot
    k_j := arg max_{k≥j} |a_kj|
    if a_{k_j j} ≠ 0 then
        # Swap rows
        A[k_j,:] ↔ A[j,:]
        piv[k_j] ↔ piv[j]
        # Elimination
        for k = j+1,...,n do
            ℓ_kj := a_kj / a_jj
            a_kj = ℓ_kj
            for i = j+1,...,n do
                a_ki = a_ki − ℓ_kj a_ji
            end
        end
    end
end
```

Algorithm 3: In-place Gaussian Elimination with Row Pivoting for implementing Factorization (same as above just without the b vector.)

1. Implement a function `solve_tri(A, b, lower=False)` which takes as input a triangular matrix A, a vector b and an optional boolean parameter `lower`, which is set to `False` by default. This function shall first check if the dimensions of the input parameters fit and if the matrix is invertible. If these two conditions are true, it shall compute and output the solution x by applying the above derived formulas.
2. Test your routine on some examples with lower and upper triangular matrices. Find the corresponding SciPy Routine and compare.

Ex 64 Linear Systems, Factor and Solve, Python

Solving Linear Systems using QR Decomposition: Factorize and Solve
[Diese Aufgabe nicht mehr stellen, da qr-factor ja schon gemacht und qr-solve exakt solve-triangular ist daher ebenfalls erledigt. daher das besser in die prog aufgabe "curve-fitting mit qr" gesetzt!]

Let $A \in \mathbb{R}^{m \times n}$ be a matrix with $n \leq m$ and linearly independent columns (this implies R is invertible) and let $b \in \mathbb{R}^m$. Then, using a QR decomposition $A = QR$, we can compute the solution x of $Ax = b$ (basically in two steps) by solving

$$Rx = Q^T b.$$

Tasks:

1. Implement a function `factor_qr(A)` which computes a reduced QR decomposition of a matrix $A \in \mathbb{R}^{m \times n}$. Thus, it shall output an orthogonal matrix $Q \in \mathbb{R}^{m \times n}$ and an upper triangular matrix $R \in \mathbb{R}^{n \times n}$, so that $A = QR$.
   You can copy the Gram-Schmidt algorithm implemented as a function `QR(A)` *from previous sheets or find an appropriate SciPy Routine.*
2. Implement a function `solve_qr((Q, R), b)` which takes as input the matrices $Q \in \mathbb{R}^{m \times n}$ and $R \in \mathbb{R}^{n \times n}$ computed by `factor_qr(A)` in form of a tuple, as well as a vector $b \in \mathbb{R}^m$. It shall then apply the solving procedure above and output the solution x of $Ax = b$.
   You can recycle the function `solve_tri(A, b, lower=False)` *from previous sheets or use an appropriate SciPy routine for triangular systems.*
3. Test your routine on multiple examples.

Ex 65

---

extended qr factor and solve code

Ex 66

Oblique Projector
For $\alpha \in \mathbb{R}$ consider the matrix

$$P_\alpha = \begin{pmatrix} 1 & \alpha \\ 0 & 0 \end{pmatrix}.$$

1. Show that P_α is a projector for all $\alpha \in \mathbb{R}$.
2. Determine $(I - P_\alpha)$ and show that it is a projector for all $\alpha \in \mathbb{R}$.
3. Find a basis for $\mathsf{Im}(P_\alpha)$ and $\ker(I - P_\alpha)$ as well as $\ker(P_\alpha)$ and $\mathsf{Im}(I - P_\alpha)$.
4. For an arbitrary $y \in \mathbb{R}^2$ determine a vector $v = v(y, \alpha) \in \mathsf{Im}(P_\alpha)$ and a vector $r = r(y, \alpha) \in \ker(P_\alpha)$, so that $y = v + r$. Are v and r unique?
5. Determine the α for which P_α is an orthogonal projector.

Ex 67

Classical Gram–Schmidt Algorithm
Let $A \in \mathbb{R}^{n \times n}$ be defined as

$$A := \begin{pmatrix} 3 & 4 \\ 2 & 7 \end{pmatrix}.$$

1. Compute a QR-decomposition of A using the Gram-Schmidt algorithm.
2. Compute $QR = A$ to check your result.

Ex 68

Consider the matrix

$$A := \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix}.$$

1. Compute a QR-decomposition of A using the Gram–Schmidt algorithm.
   (*Hint: Verify the desired properties of the factor matrices and test* $QR = A$.))
2. Is A invertible? Use your QR decomposition to explain your answer.

Ex 69

QR Factorization
Let $A = \widehat{Q}\widehat{R}$ be a reduced QR factorization of $A \in \mathbb{R}^{m \times n}$. Show that

$$\ker(A) = \{0\} \Leftrightarrow \widehat{R} \in \mathsf{GL}_n(\mathbb{R}).$$

Ex 70

Solving Linear Systems using the QR Decomposition
Let $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$ be given. Assume you already have the QR decomposition of A, i.e., an orthogonal matrix Q and an upper triangular matrix R, so that $A = QR$. Further assume that R has *nonzero* diagonal elements (i.e., $r_{ii} \neq 0$).

1. How can you use the QR decomposition $A = QR$ for solving a linear system of the form

$$Ax = b \ \ ?$$

2. Use your idea from 1. to solve the system $Ax = b$ where

$$A := \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} = QR, \quad \text{with} \quad Q = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{3}} \\ 0 & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{3}} \end{pmatrix}, \ R = \begin{pmatrix} \frac{2}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & \frac{3}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ 0 & 0 & \frac{2}{\sqrt{3}} \end{pmatrix} \quad \text{and} \quad b := \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

   Remark: The square root terms will cancel out nicely while solving for x.

Solving Linear Systems with Triangular Matrices: Forward/Backward Substitution

1. Solve the following linear system by hand:

$$\begin{pmatrix} 1 & -1 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix}.$$

   Hint: Start with the last equation/row.

2. Let $U = (u_{ij})_{ij} \in \mathbb{R}^{n \times n}$ be an upper triangular matrix, i.e., $u_{ij} = 0$ for $i > j$ and let $b \in \mathbb{R}^n$ be a given vector.
   a) Derive a general (iterative) formula to obtain $x \in \mathbb{R}^n$, which solves $Ux = b$.
   b) What requirements have to be put on the diagonal entries u_{ii} of U, so that the system has a unique solution?

   Hint: Write out the system as

$$\begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & & \vdots \\ \vdots & & \ddots & u_{n-1,n} \\ 0 & \cdots & 0 & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}$$

   and start with the last row to determine x_n. Then use x_n to determine x_{n-1} from the second but last row. Continue this procedure until you reach the first row to determine x_1 with the help of the previously determined values $x_2, \ldots, x_n$.

3. Find a similar formula for lower triangular matrices $L = (\ell_{ij})_{ij} \in \mathbb{R}^{n \times n}$, for which $\ell_{ij} = 0$ for $i < j$.

Remark: A diagonal matrix is a special case of a triangular matrix.

4 Eigenvalues

Ex 72 Linear Algebra, Eigenvalues

```
def r(coeffs):
    coeffs = np.array(coeffs[:-1])
    n = len(coeffs)
    A = np.eye(n, k = -1)
    A[:,-1] = -coeffs
    lam, v = np.linalg.eig(A)
    return lam
```

1. Which algorithm is implemented in the function above? What role does the parameter `coeffs` play?
2. Which value `lam` will the function return at `coeffs = [-1,0,1]`? (No proof needed)
3. What would the function return at `coeffs = [-1,0,-1]`? Can you give a suggestion for improvement, in order to make the function more robust to bad input data?

Ex 73

Compute the eigenvalues of the following matrices.

1. $$A = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix}$$
2. $$B = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & -9 \\ 0 & 1 & 6 \end{pmatrix}$$
3. $$C = \begin{pmatrix} \pi & 3 & -1 & 6 \\ 0 & 4 & 1 & 7 \\ 0 & 0 & i & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{pmatrix}$$

Ex 74

Compute the eigenvalues of the following matrices.

1. $$A = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$$
2. $$B = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 3 \\ 2 & 0 & 1 \end{pmatrix}$$
3. $$C = \begin{pmatrix} \pi & 0 & 0 & 0 & 0 \\ 1 & -7 & 0 & 0 & 0 \\ 2 & 0 & i & 0 & 0 \\ 3 & 0 & 8 & 0 & 0 \\ 4 & 7 & 9 & 0 & \frac{1}{2} \end{pmatrix}$$

Ex 75

Compute the eigenvalues of the following matrices.

1.
$$A = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix}$$

2.
$$B = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 4 \\ 0 & 1 & 1 \end{pmatrix}$$

3.
$$C = \begin{pmatrix} \pi & 3 & -1 & 2 \\ 0 & 4 & 1 & 7 \\ 0 & 0 & i & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{pmatrix}$$

Ex 76

Compute the eigenvalues of the following matrices.

1.
$$A = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix}$$

2.
$$B = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & -9 \\ 0 & 1 & 6 \end{pmatrix}$$

Ex 77 Linear Algebra, Eigenvalues

Compute Gershgorin Disks
Compute the Gershgorin disks of the following matrices and denote the matrices which necessarily have positive eigenvalues:

$$A = \begin{pmatrix} 14 & 10 & 8 \\ 10 & 14 & 8 \\ 8 & 8 & 6 \end{pmatrix}, \quad B = \begin{pmatrix} 10 & 1 & 5 \\ 1 & 17 & 5 \\ 5 & 5 & 25 \end{pmatrix}, \quad C = \begin{pmatrix} 5 & 1 & 3 \\ 1 & -2 & -3 \\ 3 & -3 & -6 \end{pmatrix}.$$

Ex 78 Linear Algebra, Eigenvalues

Eigendecomposition
Consider the matrix
$$A := \begin{pmatrix} 2 & 3 \\ 3 & 2 \end{pmatrix} \in \mathbb{R}^{2\times 2}.$$

1. Why does this matrix possess an eigendecomposition $A = Q\Lambda Q^T$? Compute the matrices Λ and Q, by following this recipe:
   a) Determine its eigenvalues λ_1 and λ_2 to find Λ by solving $\chi_A(\lambda) = \det(A - \lambda I) = 0$.
   b) Determine the corresponding eigenvectors v_1 and v_2 by solving $(A - \lambda_i I)v = 0$.
   c) Normalize the eigenvectors to find Q by setting $\tilde{v}_i := \frac{v_i}{\|v_i\|_2}$ and $Q := [\tilde{v}_1, \tilde{v}_2]$. Test if $Q^T Q$ equals I_2.
   d) Test if $Q\Lambda Q^T$ equals A.
2. Use the eigendecomposition of A to derive its inverse A^{-1}.

Ex 79 Linear Algebra, Eigenvalues

Eigenvalue Characteristics
We denote the identity matrix of $\mathbb{F}^{n\times n}$ by I. Let $A \in \mathbb{F}^{n\times n}$ be a matrix. Some value $\lambda \in \mathbb{C}$ is called eigenvalue of A, if there is a vector $v \neq 0$ in $\mathbb{F}^n$ such that

$$(A - \lambda I)v = 0,$$

where $0 \in \mathbb{F}^n$ denotes the zero vector. Use the above definition to prove the following assertions.

1. If A is invertible, then for all eigenvalues λ of A we have $\lambda \neq 0$ and $\frac{1}{\lambda}$ is an eigenvalue of A^{-1}.
2. If λ is an eigenvalue of A, then $(\lambda - \alpha)$ is an eigenvalue of $(A - \alpha I)$ for any $\alpha \in \mathbb{C}$.
3. If λ is an eigenvalue of A, then λ is also an eigenvalue of $Q^\top AQ$ for any orthogonal matrix $Q \in \mathbb{F}^{n\times n}$.

Ex 80

Let $A \in \mathbb{F}^{n\times n}$ be invertible and λ some eigenvalue of A.

1. Please show that $\lambda \neq 0$.
2. Please show that $\frac{1}{\lambda}$ is an Eigenvalue of A^{-1}.

Ex 81

Eigenvalues and Positivity of a Matrix
Definition. A matrix $A \in \mathbb{R}^{n\times n}$ is called positive definite (semi-definite) it $x^T Ax > 0 \ \ (\geq 0)$ for all $x \in \mathbb{R}^n \setminus \{0\}$.

Let $A \in \mathbb{R}^{n\times n}$ be any matrix. Please show:

1. If A is positive definite (semi-definite), then $\lambda > 0$ (≥ 0) for all eigenvalues $\lambda \in \sigma(A)$.
   (*Hint:* Rayleigh quotient.)
2. The reverse is not true: Find an example of a matrix, which has positive eigenvalues, but is *not* positive definite.
   (*Hint:* Consider, e.g., a triangular 2×2 matrix.)
3. The reverse is true for symmetric matrices: Let A be *symmetric* and $\lambda > 0$ (≥ 0) for all eigenvalues $\lambda \in \sigma(A)$, then A is positive definite (semi-definite).
   (*Hint:* Eigendecompostion.)
4. Let A be *symmetric*, then A is invertible if and only if $\lambda \neq 0$ for all eigenvalues $\lambda \in \sigma(A)$.
   (*Hint:* Eigendecompostion.)

Ex 82

Eigenvalues and Positivity of a Matrix
Definition. A symmetric matrix $A \in \mathbb{R}^{n\times n}$ is called positive definite (semi-definite) if $x^T Ax > 0 \ \ (\geq 0)$ for all $x \in \mathbb{R}^n \setminus \{0\}$.

Let $A \in \mathbb{R}^{n\times n}$ be any matrix. Please show:

1. Find an example of a nonsymmetric matrix, which has strictly positive eigenvalues, but $x^T Ax < 0$ for some $x \in \mathbb{R}^n$.
   (*Hint:* Consider, e.g., a triangular 2×2 matrix.)
2. Let A be *symmetric* and $\lambda > 0$ (≥ 0) for all eigenvalues $\lambda \in \sigma(A)$, then A is positive definite (semi-definite).
3. Show that symmetric and positive definite matrices are invertible.

Ex 83

Properties of Eigenvalues
Prove the following statements (see also the corresponding Lemma from the lecture):

1. *The eigenvalues of the powers of a matrix:*
   Let $A \in \mathbb{F}^{n\times n}$, $\lambda \in \sigma(A)$ then $\lambda^k \in \sigma(A^k)$ for any $k \in \mathbb{N}$.
2. *Eigenvalues of invertible Matrices:*
   Let $A \in \mathbb{F}^{n\times n}$ be invertible and $\lambda \in \sigma(A)$, then $\lambda \neq 0$ and $\frac{1}{\lambda}$ is an eigenvalue of A^{-1}.
3. *Eigenvalues of a scaled matrix:*
   Let $A \in \mathbb{F}^{n\times n}$ and $\lambda \in \sigma(A)$, then $\alpha\lambda \in \sigma(\alpha A)$ for any $\alpha \in \mathbb{F}$.
4. *Real symmetric matrices have real eigenvalues:* (Not obvious without using properties of complex numbers! Solution not relevant for exam and thus not discussed here.)
   $A \in \mathbb{R}^{n\times n}$, $A = A^T \ \Rightarrow \ \sigma(A) \subset \mathbb{R}$.

5. *The eigenvalues of real orthogonal matrices:* (Not obvious without using properties of complex numbers! Solution not relevant for exam and thus not discussed here.)
   $Q \in \mathbb{R}^{n\times n}$ be orthogonal, $\lambda = a + ib \in \sigma(Q) \quad \Rightarrow \quad |\lambda| := \sqrt{a^2 + b^2} = 1$
6. *The eigenvalues of an upper (or lower) triangular matrix are sitting on its diagonal:*
   Let $U \in \mathbb{F}^{n\times n}$ with $u_{ij} = 0$ for $i > j$. Then $\sigma(U) = \{u_{11}, \ldots, u_{nn}\}$.
7. *Similar matrices have the same eigenvalues:*
   Let $A \in \mathbb{F}^{n\times n}$ and $T \in GL_n(\mathbb{F})$, i.e., T is invertible. Then $\sigma(A) = \sigma(T^{-1}AT)$.
8. *Eigenvalues of a shifted matrix:*
   Let $A \in \mathbb{F}^{n\times n}$ and $\lambda \in \sigma(A)$, then $(\lambda - \alpha)$ is an eigenvalue of $(A - \alpha I)$ for any $\alpha \in \mathbb{F}$.
9. *Symmetric matrices have orthogonal eigenvectors:* (Solution not relevant for exam and thus not discussed here.)
   Let $\lambda_1 \neq \lambda_2$ be two *distinct* eigenvalues of a real *symmetric* matrix $A \in \mathbb{R}^{n\times n}$ (i.e., $A = A^T$), and let $v_1, v_2 \in \mathbb{R}^n$ be corresponding eigenvectors. Proof that v_1 and v_2 are orthogonal, i.e., $v_1^\top v_2 = 0$.

Ex 84

Properties of Eigenvalues
Prove the following statements (see also the corresponding Lemma from the lecture):

1. *The eigenvalues of the powers of a matrix:*
   Let $A \in \mathbb{F}^{n\times n}$, $\lambda \in \sigma(A)$ then $\lambda^k \in \sigma(A^k)$ for any $k \in \mathbb{N}$.
2. *Eigenvalues of invertible Matrices:*
   Let $A \in \mathbb{F}^{n\times n}$ be invertible and $\lambda \in \sigma(A)$, then $\lambda \neq 0$ and $\frac{1}{\lambda}$ is an eigenvalue of A^{-1}.
3. *Eigenvalues of a scaled matrix:*
   Let $A \in \mathbb{F}^{n\times n}$ and $\lambda \in \sigma(A)$, then $\alpha\lambda \in \sigma(\alpha A)$ for any $\alpha \in \mathbb{F}$.
4. *The eigenvalues of an upper (or lower) triangular matrix are sitting on its diagonal:*
   Let $U \in \mathbb{F}^{n\times n}$ with $u_{ij} = 0$ for $i > j$. Then $\sigma(U) = \{u_{11}, \ldots, u_{nn}\}$.
5. *Eigenvalues of a shifted matrix:*
   Let $A \in \mathbb{F}^{n\times n}$ and $\lambda \in \sigma(A)$, then $(\lambda - \alpha)$ is an eigenvalue of $(A - \alpha I)$ for any $\alpha \in \mathbb{F}$.
6. *Hermitian matrices have orthogonal eigenvectors:*
   Let $\lambda_1 \neq \lambda_2$ be two *distinct* eigenvalues of a *Hermitian* matrix $A \in \mathbb{C}^{n\times n}$ (i.e., $A = A^H$), and let $v_1, v_2 \in \mathbb{C}^n$ be corresponding eigenvectors. Proof that v_1 and v_2 are orthogonal, i.e., $v_1^H v_2 = 0$.

Ex 85

Let

$$Q := \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

1. What can you say about the columns of Q? Justify your answer rigorously.
2. Compute the determinant of Q and the eigenvalues of Q.
3. Interpret the function $\mathbb{R}^3 \to \mathbb{R}^3, x \mapsto Qx$ geometrically.

Ex 86

Please show that the eigenvalues of an upper triangular matrix $A \in \mathbb{R}^{n\times n}$ are given on its diagonal.

Ex 87

Eigenvalues Triangular Matrix
Use Theorem 12.2iii) to show that the eigenvalues of an upper triangular matrix $A \in \mathbb{R}^{n\times n}$ are given on its diagonal, i.e., that $\det(A - \lambda I) = 0$ for all $\lambda \in \mathit{diag}(A)$.

Ex 88 Linear Algebra, Eigenvalues

Gershgorin Disks
Let $A \in \mathbb{C}^{n\times n}$ be a matrix with entries a_{ij} for $i, j = 1, \ldots, n$. Let $R_i := \sum_{j\neq i} |a_{ij}|$ be the sum of the absolute values of the non-diagonal entries in the i-th row. Moreover, let

$$D(a_{ii}, R_i) := \{z \in \mathbb{C} \,|\, \|z - a_{ii}\| \leq R_i\}$$

be the disk of radius R_i and centre a_{ii} in $\mathbb{C}$. Prove the following theorem.

Theorem: Every eigenvalue of A lies within at least one of the Gershgorin disks $D(a_{ii}, R_i)$, i.e., $\forall\ \lambda \in \sigma(A)\ \exists\ i \in \{1, \ldots, n\}:\quad \lambda \in D(a_{ii}, R_i)$.

Ex 89 Linear Algebra, Eigenvalues

Inverse Power Method
Let $A \in \mathbb{R}^{n\times n}$ be a matrix with eigenvalues λ_j for $j \in \{1, \ldots, n\}$ with the property

$$|\lambda_n| \geq \cdots \geq |\lambda_{i+1}| > |\lambda_i| > |\lambda_{i-1}| \geq \cdots \geq |\lambda_1|.$$

Let $\hat{\lambda} \approx \lambda_i$ be a *good guess* for the eigenvalue λ_i, which means

$$0 < |\lambda_i - \hat{\lambda}| < |\lambda_j - \hat{\lambda}| \quad \forall\ i \neq j. \tag{1}$$

Or in other words, $\hat{\lambda}$ is closer to λ_i (in absolute terms) than any other eigenvalue.
Let us define

$$B := (A - \hat{\lambda} I).$$

1. Show that $(\lambda_i - \hat{\lambda})^{-1}$ is (in magnitude) the largest eigenvalue of the matrix B^{-1}.
2. Let $x^0 \in \mathbb{R}^n$. With respect to A, the iteration

$$x^{k+1} := \frac{B^{-1}x^k}{\|B^{-1}x^k\|} \tag{2}$$

for $k \geq 0$ is called *inverse power iteration*. Under which condition on x^0 does this iteration converge and what is the limit?

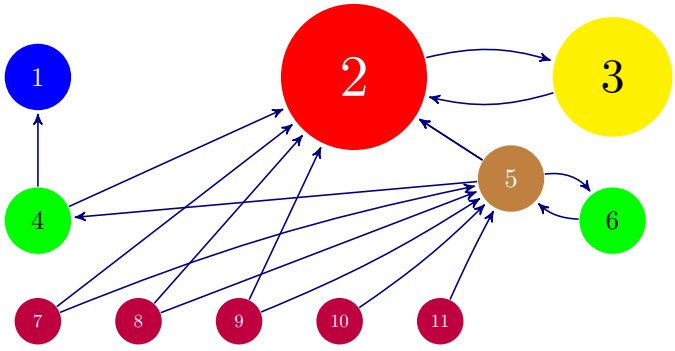
Hint: Apply the theorem about the power method from the lecture.

An Application of Eigenvectors: The *PageRank*
Aim: To rank results of a web search engine (such as Google) according to the *"importance"* of the web pages.

1998: For this purpose, Larry Page and Sergei Brin developed the PageRank algorithm as the basis of the Google empire.

Assumption: *"important"* means more links from other (important) web pages.

Idea: Let us think of the web as a directed graph, i.e., web pages are nodes and links from one page to another, i.e, from one node to another, are modeled as directed edges. For example a web structure consisting of 11 web pages could look as follows:



We now randomly place a random surfer according to the initial probability distribution $x^0 = (x_1^0, \ldots, x_n^0)$ on this graph. Here, n (in the example above $n = 11$) denotes the number of web pages and x_i^0 denotes the probability that the random surfer *starts* at web page i. Further let $e := (1, \ldots, 1)^T$, then the fact that x^0 is a probability distribution (i.e., probabilities sum up to 1) translates into $e^T x^0 = x_1^0 + \ldots + x_n^0 = 1$. Now we make the assumption that the random surfer moves...

(1) ...with probability $\alpha \in (0, 1)$ according to the link structure (with equal preferences to outgoing links)

(2) ...with probability $(1 - \alpha)$ he can teleport to a random page (with equal probability) to prevent stranding in deadlocks

→ Pages, where the random surfer is more likely to appear in the long run based on the web's structure are considered more important.

These two movements can be described by multiplying the current probability distribution with the two following matrices:

$$P_1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{array} \\ \begin{pmatrix} 1 & & & 1/2 & & & & & & & \\ & & 1 & 1/2 & 1/3 & & 1/2 & 1/2 & 1/2 & & \\ & 1 & & & & & & & & & \\ & & & & 1/3 & & & & & & \\ & & & & & 1 & 1/2 & 1/2 & 1/2 & 1 & 1 \\ & & & & 1/3 & & & & & & \\ & & & & & & & & & & \\ & & & & & & & & & & \\ & & & & & & & & & & \\ & & & & & & & & & & \\ & & & & & & & & & & \end{pmatrix} \end{array}, \quad P_2 := \frac{1}{n} e e^T = \left(\frac{1}{n}\right)_{ij}$$

More precisely,

(1) **Link structure:** P_1 is the probability matrix (column stochastic) defined by
P_1^{ij} := Probability that random surfer moves from page j to page i defined by the link structure

(2) **Jumps:** P_2 is the probability matrix (column stochastic) defined by
$P_2^{ij} := \frac{1}{n}$= Probability that random surfer jumps from page j to page i

The movement of the random surfer is then completely defined by the probability matrix

$$P = \alpha P_1 + (1 - \alpha) P_2.$$

This matrix is also known as the ***Google Matrix***. For the next time instances we therefore obtain

$$\begin{aligned} x^1 &= \alpha P_1 x^0 + (1 - \alpha) P_2 x^0 = P x^0 \\ x^2 &= \alpha P_1 x^1 + (1 - \alpha) P_2 x^1 = P x^1 \\ x^{k+1} &= \alpha P_1 x^k + (1 - \alpha) P_2 x^k = P x^k = P^{k+1} x^0 \\ x^* &= \lim_{k \to \infty} x^k =: \textbf{\textit{PageRank}} \end{aligned}$$

Observations:

- One can easily show that P_1, P_2 and thus P are column stochastic (i.e., $e^T P = e^T$)
- Consequently, since x^0 is a probability distribution (i.e., $e^T x^0 = 1$), also $e^T x^k = 1$ for all k and $e^T x^* = 1$

Question:
Does this sequence $\{x^k\}_{k \in \mathbb{N}}$ of vectors converge (to a steady state)? More precisely, is there a $x^* = \lim_{k \to \infty} x^k = \lim_{k \to \infty} P^k x^0$, so that

$$P x^* = 1 x^*. \tag{1}$$

→ With other words, is there an ***eigenvector*** x^* to the ***eigenvalue*** 1 of the matrix P?

→ ***Eigenvalue algorithms*** are developed to solve such problems. One of them is the ***Power iteration***, which, applied to the eigenvalue problem above, produces precisely the sequence

$$x^k = P^k x^0.$$

Intuitively, in the limit most of the "mass" would be located at web pages that have many incoming links and would therefore be ranked as being more important. In fact, the i-th component of x^* is called the *PageRank* of the web page i.

Remark: *Perron Theorem*
A positive damping factor $\alpha > 0$ is also technically necessary as it assures that the matrix P has only strictly positive coefficients. The Perron Theorem then sates that its largest eigenvalue is strictly larger than all other eigenvalues (in magnitude). Thus the convergence of the Power method is guaranteed. Since the matrix is column stochastic one can further show that the largest eigenvalue is 1.

Ex 90 Linear Algebra, Eigenvalues, Python

The Power Iteration and the *PageRank*

1. Implement a function `power_iteration(A,m,p=1)` which takes as input a matrix $A \in \mathbb{R}^{n\times n}$, a maximum iteration number $m \in \mathbb{N}$ and an *optional* parameter p which determines the order of the p-norm and is set to $p = 1$ by default. This function shall then return the m-th iterates x_m and μ_m of the power iteration.

   Hints:

   - You can use a random distribution x_0 as initial guess by calling for example the numpy function

     ```
     x = numpy.random.dirichlet(np.ones(n),size=1).reshape(n)
     ```

     or simply choose

     ```
     x = 1./n * np.ones(n).
     ```

   - For the normalization step use the numpy function

     ```
     numpy.linalg.norm(x, ord=p),
     ```

     which allows the choices $p \in \{1, 2, \infty\}$ (among others).

2. Determine the **PageRank** of the web structure given above. Therefore apply your function `power_iteration(A,m,p=1)` to the PageRank matrix

   $$A := P = \alpha P_1 + (1-\alpha)P_2,$$

   where

   $$P_1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{ccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{array} \\ \begin{pmatrix} 1 & & & 1/2 & & & & & & & \\ & & 1 & 1/2 & 1/3 & & 1/2 & 1/2 & 1/2 & & \\ & 1 & & & & & & & & & \\ & & & & 1/3 & & & & & & \\ & & & & & 1 & 1/2 & 1/2 & 1/2 & 1 & 1 \\ & & & & 1/3 & & & & & & \\ & & & & & & & & & & \\ & & & & & & & & & & \\ & & & & & & & & & & \\ & & & & & & & & & & \\ & & & & & & & & & & \end{pmatrix} \end{array}, \quad P_2 := \frac{1}{n} e e^T = \left(\frac{1}{n}\right)_{ij}.$$

   Play around with the damping factor α. What do you observe?

   Hint: For implementing P_1 and P_2, and thus P, you can use this code snippet.

```
import numpy as np
def P(alpha):
    P_1=np.array([[1,0,0,1.0/2,0,0,0,0,0,0,0],
                  [0,0,1.0,1.0/2,1.0/3,0,1.0/2,1.0/2,1.0/2,0,0],
                  [0,1.0,0,0,0,0,0,0,0,0,0],
                  [0,0,0,0,1.0/3,0,0,0,0,0,0],
                  [0,0,0,0,0,1.0,1.0/2,1.0/2,1.0/2,1.0,1.0],
                  [0,0,0,0,1.0/3,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0]])
```

```
    P_2 = (1.0 / 11.0) * np.ones((11,11))

    return alpha * P_1 + (1-alpha) * P_2
```

Ex 91 Linear Algebra, Eigenvalues, Python

The QR Algorithm

1. Implement the QR eigenvalue algorithm as a function `eig(A,m)`. The function shall take as input a matrix $A \in \mathbb{R}^{n \times n}$ and a maximum iteration number $m \in \mathbb{N}$. It shall return the diagonal of the last iterate A_m. For the QR decomposition you can use the Gram-Schmidt algorithm from previous sheets or an appropriate Python function.

2. Test your algorithm on a random matrix $A \in \mathbb{R}^{n \times n}$. In order to generate such a random matrix use the following code snippet:

```
def A_gen(n):
   from numpy as np
   from scipy.linalg import qr
   A = np.random.rand(n,n)
   Q, R = qr(A)
   Lambda = np.diag(np.arange(1,n+1))
   A = Q @ (Lambda @ Q.T)
return A
```

3. Find a routine in `Scipy` to compute the eigenvalues and -vectors of a matrix. Test the routine on multiple examples, especially for higher dimensions. Compare to your algorithm.

Ex 92 Linear Algebra, Determinant

Rule of Sarrus

Derive the *Rule of Sarrus* for the determinant of a (3×3)-matrix by using the Laplace formula from the lecture with $n = 3$. Then compute the determinant of

$$A = \begin{pmatrix} 1 & 4 & 3 \\ 0 & 2 & 2 \\ 0 & 1 & 1 \end{pmatrix}.$$

What does it tell us about the columns?

---

5 Singular Values and SVD

Ex 93 Linear Systems, Factor and Solve

Frobenius Matrices
Let $\ell_j := (0, \dots, 0, \ell_{j+1,j}, \dots, \ell_{m,j})^\top \in \mathbb{R}^m$, $e_j \in \mathbb{R}^m$ be the j-th unit vector and $I \in \mathbb{R}^{m\times m}$ be the identity matrix. Then show that the matrix

$$L_j := I + \ell_j e_j^\top \in \mathbb{R}^{m\times m}$$

satisfies:

1. The matrix L_j is an invertible lower triangular matrix.
2. The inverse of L_j is given by $L_j^{-1} = I - \ell_j e_j^\top \in \mathbb{R}^{m\times m}$.
3. For $i \le j$ it holds that $L_i L_j = I + \ell_j e_j^\top + \ell_i e_i^\top$ and $L_i^{-1} L_j^{-1} = I - \ell_j e_j^\top - \ell_i e_i^\top$.

Ex 94 Linear Algebra, Singular Values

Frobenius Norm, Trace and Singular Values
The Frobenius norm of a matrix $A \in \mathbb{R}^{m\times n}$ is defined by

$$\|A\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}$$

and the trace of a matrix $B \in \mathbb{R}^{n\times n}$ by

$$\mathsf{tr}(B) = \sum_{i=1}^{n} b_{ii}.$$

1. Show that for a matrix $A \in \mathbb{R}^{m\times n}$ we have
$$\|A\|_F^2 = \mathsf{tr}(A^T A).$$
2. Show that the trace is symmetric, i.e., for $A, B \in \mathbb{R}^{m\times n}$ we find
$$\mathsf{tr}(A^\top B) = \mathsf{tr}(B^\top A) = \mathsf{tr}(BA^\top).$$
Remark: $\mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}, (A, B) \mapsto \mathsf{tr}(B^\top A)$ defines an inner product on $\mathbb{R}^{m\times n}$ and the Frobenius norm $\|A\|_F = \sqrt{\mathsf{tr}(A^T A)}$ denotes the corresponding norm. For example, Cauchy-Schwarz inequality holds.
3. Use these results and the singular value decomposition to show that the Frobenius norm of a matrix $A \in \mathbb{R}^{m\times n}$ can be expressed in terms of its singular values, i.e.,
$$\|A\|_F^2 = \sum_{i=1}^{\min(m,n)} \sigma_i^2 \quad \left(= \sum_{i=1}^{r} \sigma_i^2 = \|\Sigma\|_F^2\right).$$

Ex 95 Linear Algebra, Singular Values

An ill-conditioned Diagonal Matrix
For $n \in \mathbb{N}$ consider the diagonal matrix

$$D_n = \mathsf{diag}\left(1, \frac{1}{2}, \dots, \frac{1}{n}\right) = \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{n} \end{pmatrix} \in \mathbb{R}^{n\times n}.$$

Tasks:

1. Is D_n invertible? Explain your answer.
2. For a given $b \in \mathbb{R}^n$, determine the corresponding $x_b \in \mathbb{R}^n$, such that $D_n x_b = b$. Is x_b uniquely defined? Explain your answer.
3. Determine the spectrum $\sigma(D_n)$ of D_n.

4. Find a singular value decomposition of D_n.
5. What is the condition number $\text{cond}_2(D_n)$ of D_n? Determine $\lim_{n\to\infty} \text{cond}_2(D_n)$.
6. Let us assume $b \in \mathbb{R}^n$ is the true right-hand side and $\tilde{b}$ would be our measured right-hand side, which is prone to some error. For simplicity let us assume $\tilde{b} = b + \varepsilon e$ for some fixed small $\varepsilon > 0$ and $e = (1, \dots, 1)^T \in \mathbb{R}^n$ (i.e., each component of b is equally perturbed by ε). Consider the difference $\Delta x := x_b - x_{\tilde{b}}$ and estimate the relative error $\frac{\|\Delta x\|}{\|x\|}$. What happens for large n?

Some Background

A column in A contains the measured features (e.g., age and height) for a particular sample (e.g., a person) and a row contains all measured values for a particular feature. Thus, let $a_i \in \mathbb{R}^n$ denote a row of A, then by assuming that the mean $a_i^\top \mathbf{1}$ is zero (without loss of generality, otherwise center the data) for all features, we have

$$\frac{1}{n-1} a_i^\top a_j = \begin{cases} \text{“statistical variance in feature } i\text{”} & i = j \\ \text{“statistical covariance between feature } i \text{ and } j\text{”} & i \neq j. \end{cases}$$

Furthermore we observe that the matrix

$$\frac{1}{n-1} AA^\top = \frac{1}{n-1} \left(a_i^\top a_j\right)_{ij}$$

contains these covariances (it is therefore often called *sample covariance matrix*) and using the SVD $A = U\Sigma V^\top$ we find

$$\frac{1}{n-1} AA^T = \frac{1}{n-1} U \begin{pmatrix} \sigma_1^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_r^2 \end{pmatrix} U^T = \frac{1}{n-1} \sum_{i=1}^{r} \sigma_i^2 u_i u_i^T.$$

The first few summands explain most of $\frac{1}{n-1} AA^T$, i.e., the sample covariance matrix, and the singular vectors $u_1, \dots, u_r$ are called principal components.

Geometrically we have the following interpretation:

$$A = \begin{matrix} m \text{ feats} \\ \downarrow \end{matrix} \begin{pmatrix} | & & | & & | \\ a_1 & \cdots & a_i & \cdots & a_n \\ | & & | & & | \end{pmatrix} \begin{matrix} \\ \longrightarrow \\ n \text{ samples} \end{matrix} = U\Sigma V^T = \underbrace{\begin{pmatrix} | & & | \\ u_1 & \cdots & u_m \\ | & & | \end{pmatrix}}_{\text{orthonormal basis}} \underbrace{(\Sigma V^T)}_{\text{coordinates of } a_i \text{ in terms of this basis}}$$

Thus, each sample $a_i \in \mathbb{R}^m$ is a linear combination of $u_1, \dots, u_m$ with coefficients $(\Sigma V^T)_i$.

Relation to “total least squares”: The least squares problem

$$\min_{x \in \mathbb{R}} \|Ax - b\|^2,$$

where we want to minimize the error in the *dependent* variables, can be reformulated as the constrained optimization problem

$$\min_{r, x \in \mathbb{R}} \|r\|^2$$
$$s.t. \;\; r = Ax - b.$$

If we also want to encounter errors in the *independent* variables we arrive at the problem

$$\min_{r, s, x \in \mathbb{R}} \| \begin{pmatrix} r \\ s \end{pmatrix} \|^2$$
$$s.t. \;\; (A + s)x = b + r.$$

This problem is called the total *least squares problem* and errors on both dependent and independent variables are considered. One can show that the solution of this problem is the low-rank approximation which we yield from cropping the singular value decomposition. See for details: https://eprints.soton.ac.uk/263855/1/tls_overview.pdf

Minimum Norm Least Squares with Pseudoinverse
Let $A \in \mathbb{R}^{m\times n},\ b \in \mathbb{R}^m$. Consider the SVD $A = U\Sigma V^\top$ and set $A^+ = V\Sigma^+ U^\top$, where $\Sigma^+ = \text{diag}(\frac{1}{\sigma_1}, \dots, \frac{1}{\sigma_r}, 0, \dots, 0)$ is the pseudoinverse of a diagonal matrix as derived in the lecture. Show that

$$x^+ := A^+ b = \arg \min_{x \in \{x:\ A^\top A x = A^\top b\}} \|x\|_2^2,$$

i.e., x^+ is the minimum norm least squares solution.
Hint: First consider the simple case that A is diagonal and then use the SVD for the general case.

Ex 97 Linear Algebra, Singular Values, Python

Computation of the SVD

1. Assume you can only solve eigenvalue problems. Implement a Python function `U,V,sigma = svd(A)`, which computes a reduced SVD of some matrix $A \in \mathbb{R}^{m\times n}$ by using `numpy.linalg.eigh(A)`.
2. Test your routine on some examples.

Ex 98 Linear Algebra, Singular Values, Python

Using the SVD for Image Compression
Use the code snippet below (or an appropriate Python library) to load an image of your choice (extension `.png` or `.jpg`) as a gray scaled image $A \in \mathbb{R}^{m\times n}$.

1. Find a `scipy` routine to compute the SVD $U\Sigma V^T = A$.
2. Plot the singular values.
3. For several $1 \le k \le \text{rank}(A)$:
   a) Compute the *truncated SVD*: Use only the first k columns of U, the first k singular values $\sigma_1, \dots, \sigma_k$ from Σ and the first k rows of V^T to reconstruct A and plot the resulting image A_k using `plt.imshow(A_k, cmap='gray')`.
   b) For each k, compute the total number of floats that need to be stored for the truncated SVD A_k and compare it to the total number of floats that need to be stored for the full image A.

```
    """
    import matplotlib
    img = matplotlib.image.imread(path_to_image)
    print(np.shape(img))
    # ITU-R 601-2 luma transform (rgb to gray)
    img = np.dot(img, [0.2989, 0.5870, 0.1140])
    return img
```

Ex 99 Linear Algebra, Singular Values

Equivalent Definitions of the Matrix Rank
Let $A \in \mathbb{R}^{m\times n}$, then use the SVD $A = U\Sigma V^\top$ to show that the following statements are equivalent:

i) The maximum number of linearly independent columns of A is r.
ii) The dimension of the image of A is r, i.e., $\dim(\text{im}(A)) = r$.
iii) The number of positive singular values of A is r.

As a consequence (since $A^\top = V\Sigma^\top U^\top$), we find $\text{rank}(A) = r = \text{rank}(A^\top)$, i.e., "column rank = row rank". The dimension formulas for A and $A^\top$ then read as

$$n = \text{rank}(A) + \text{nullity}(A) = \text{rank}(A^\top) + \text{nullity}(A),$$
$$m = \text{rank}(A^\top) + \text{nullity}(A^\top) = \text{rank}(A) + \text{nullity}(A^\top).$$

Ex 100

Consider the matrix

$$A = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 1 & 1 \end{pmatrix}.$$

1. Derive its singular value decomposition (SVD).
   (*Hint:* Compute $U\Sigma V^T$ to check your result.)
2. Write A as a sum of rank-1 matrices by using the singular values and vectors.
3. Is A invertible? Use the SVD to answer this question.

Ex 101 Linear Algebra, Singular Values

Computation of the SVD
For a matrix $A \in \mathbb{R}^{m\times n}$ define

$$S := \begin{pmatrix} 0 & A \\ A^\top & 0 \end{pmatrix}.$$

1. Why does S only have real eigenvalues?
2. Determine all eigenpairs of S. Explain your answer.
3. Derive the reduced SVD of A from the eigenpairs of S.

Ex 102 Linear Algebra, Singular Values

Compute the SVD
Consider the matrix

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

i) Compute its SVD $A = U\Sigma V^T$.

ii) Write A as a sum of rank-1 matrices.

iii) Is A invertible?

Hint: For i) follow this recipe:

1. Compute the eigenvalues λ_i with eigenvectors v_i of A^TA. Index the eigenvalues so that $\lambda_1 \geq \cdots \geq \lambda_r > 0$, where $r :=$"number of positive eigenvalues". Normalize the eigenvectors v_i.
2. For $i = 1, \ldots, r$: Set $\sigma_i := \sqrt{\lambda_i}$ and $u_i := \frac{1}{\sigma_i} Av_i$. [Until here we will already have the reduced SVD]
3. Extend the bases:
   - If $r < n$: Find orthonormal $v_{r+1}, \ldots, v_n \in \ker(A)$ by solving $Av_i = 0$ and orthogonalizing.
   - If $r < m$: Find orthonormal $u_{r+1}, \ldots, u_m \in \ker(A^T)$ by solving $A^Tu_i = 0$ and orthogonalizing.

Ex 103 Linear Algebra, Singular Values

The SVD and the Rank of a Matrix
Let $A \in \mathbb{R}^{n\times n}$ with SVD $A = U\Sigma V^T$ and define rank$(A) :=$ "number of positive singular values". Show that A is invertible $\iff$ rank$(A) = n$.

Ex 104 Linear Algebra, Singular Values

The SVD of Rank-1-Matrices
Let $u \in \mathbb{R}^m \setminus \{0\}$ and $v \in \mathbb{R}^n \setminus \{0\}$ be nonzero vectors and define $A := uv^T$. Find a reduced SVD of A and shortly explain why rank$(A) = 1$.

Ex 105 Linear Algebra, Singular Values

The SVD of Symmetric Matrices
Let $A \in \mathbb{R}^{n\times n}$ be symmetric with eigenvalues $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$.

1. What are the singular values of A?
2. How does the SVD look like? What can you say about the SVD if in addition A is positive definite?
3. Give a representation of the condition number and the rank of A in terms of its eigenvalues.

---

6 Iterative Methods

Ex 106

```python
def fun(A,b, m=50):
    n = A.shape[1]
    x = np.zeros(n)
    N = 1/A.diagonal()
    for k in range(m):
        x = x - N * (A @ x - b)
        return x
```

1. Please describe what each line of the code does (please do not write into the pseudocode).
2. Which algorithm is implemented and what is its purpose? Which role does `N` play here?

Ex 107

1. Let $A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n$ be given.
   a) General linear iterations are of the form $x_{k+1} = (I - NA)x_k + Nb$ for some invertible matrix $N \in \mathbb{R}^{n\times n}$ and $x_0 \in \mathbb{R}^n$. Give a sufficient criteria for the convergence of the sequence $(x_k)_k$.
   b) How is the Jacobi iteration without relaxation defined?
   c) What is the purpose of the Jacobi iteration?
2. Assume you want to solve the system $Ax = b$, where
$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$
   a) Determine the iteration matrix $(I - NA)$ of the Jacobi iteration (without relaxation) in this example.
   b) Perform 3 iteration steps of the Jacobi iteration with initial guess $x_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, i.e., compute the iterates x_1, x_2, x_3.
   c) Does the Jacobi iteration converge here? Justify your answer.
   (*Hint: Look at task a.i again.*)

Ex 108

Weighted Jacobi, Gauß–Seidel and Sucessive Over–Relaxation
Let $A \in \mathbb{R}^{n\times n}$ be a matrix with nonzero diagonal entries $a_{ii} \neq 0$ and consider the splitting $A = L + D + U$ into lower triangular, diagonal and upper triangular part of A. Also recall that splitting methods are of the form
$$x^{k+1} = (I - NA)x^k + Nb,$$
where the matrix $M := I - NA$ is called iteration matrix.

Show the following:

1. **Weighted Jacobi:** $N = \theta D^{-1}$
   - For the iteration matrix we find
$$M_{Jac} := I - \theta D^{-1}A = (1-\theta)I - \theta D^{-1}(L+U)$$
   - The i-the component of $x^{k+1} = (I - NA)x^k + Nb$ is given by
$$x_i^{k+1} = (1-\theta)x_i^k + \frac{\theta}{a_{ii}}\left(b_i - \sum_{j\neq i} a_{ij}\, x_j^{k+1}\right).$$
2. **Gauß–Seidel:** $N = (L + D)^{-1}$

- For the iteration matrix we find

$$M_{GS} := I - (L+D)^{-1}A = -(L+D)^{-1}U$$

- The i-the component of $x^{k+1} = (I - NA)x^k + Nb$ is given by

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}\, x_j^k \right).$$

3. **Sucessive Over–Relaxation (variant of Gauß-Seidel):** $N = \theta \cdot (\theta L + D)^{-1}$
   - For the iteration matrix we find

$$M_{SOR} := I - \theta(\theta L + D)^{-1}A = (\theta L + D)^{-1}((1-\theta)D - \theta U).$$

   - The i-the component of $x^{k+1} = (I - NA)x^k + Nb$ is given by

$$x_i^{k+1} = (1-\theta)x_i^k + \frac{\theta}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}\, x_j^k \right).$$

Remark: We observe that SOR for $\theta = 1$ is Gauß–Seidel and otherwise is a combination of the previous step x^k and the Gauß-Seidel update. For spd matrices it allows for $\theta > 1$ which is why it is called over–relaxation.

Hint: For 2. and 3. cast the formulas into the form $x^{k+1} = T^{-1}w$ for some lower triangular matrix T and some vector w and then use forward substitution.

Ex 109 Linear Systems, Krylov Subspace Methods

Arnoldi and Lanczos Iteration
Let $A \in \mathrm{GL}_n(\mathbb{R})$ and $b \in \mathbb{R}^n \setminus \{0\}$. Then consider the Arnoldi iteration as sketched in Algorithm 4 to produces an orthonormal basis $q_1, \ldots, q_r$ of the r-th Krylov subspace $K_r(A,b)$ with $r \leq \max_{s \leq n} \dim(K_s(A,b))$. Further let $Q_r := [q_1, \ldots, q_r] \in \mathbb{R}^{n \times r}$ and $H_r := Q_r^T A Q_r \in \mathbb{R}^{r \times r}$.

1. In the j-th step: Assume $q_1, \ldots, q_j$ have been computed according to the Arnoldi iteration 4 and assume that $q_1, \ldots, q_{j-1}$ are mutually orthonormal. Show that q_j is orthogonal to all $q_1, \ldots, q_{j-1}$.
2. Derive an expression for the (ℓ, k)-th entries of H_r and find these numbers in the Arnoldi iteration. What structure does H_r have?
3. Now assume A is symmetric. How does H_r look in this case? How can you simplify the Arnoldi iteration?
4. How do the eigenvalues of H_n and A relate? Explain your answer.

INPUT: $A \in GL_n(\mathbb{R})$, $b \in \mathbb{R}^n$, $r \leq n$
OUTPUT: orthonormal basis $q_1, \ldots, q_r$ of the r-th Krylov subspace $K_r(A,b)$

$q_1 := \frac{b}{\|b\|_2}$
for $j = 2, ..., r$ **do**
  $\widehat{q}_j := Aq_{j-1} - \sum_{\ell=1}^{j-1} q_\ell^\top (Aq_{j-1}) \cdot q_\ell$
  if $\|\widehat{q}_j\|_2 = 0$ **then**
    break
  end
  $q_j := \frac{\widehat{q}_j}{\|\widehat{q}_j\|_2}$
end

Algorithm 4: Arnoldi Iteration

Ex 110 Linear Systems, Splitting Methods

Convergence Speed of Linear Iterations
Let $M \in \mathbb{R}^{n\times n}$ be *symmetric* with $\rho(M) < 1$, let $N \in \mathbb{R}^{n\times n}$ and $x_0, b \in \mathbb{R}^n$. Consider the fixed point iteration

$$x_{k+1} = Mx_k + Nb$$

and show the following convergence result

$$\|x_k - x^*\|_2 \leq \rho(M)^k \|x_0 - x^*\|_2,$$

where x^* is the associated fixed point. Thus, the smaller the spectral radius, the faster does the method converge.
Hint: For symmetric matrices $M \in \mathbb{R}^{n\times n}$ we have $\|Mx\|_2 \leq \rho(M)\|x\|_2$ for all $x \in \mathbb{R}^n$.

Ex 111 Linear Systems, Krylov Subspace Methods, Python

GMRES

1. **Arnoldi step:** For given orthonormal vectors $q_1, \ldots, q_{r-1} \in \mathbb{R}^n$ considered as a matrix $Q_{r-1} = [q_1, \ldots, q_{r-1}] \in \mathbb{R}^{n\times(r-1)}$, and a vector $v \in \mathbb{R}^n$, implement a helper function

   $$q_r, h_r := \texttt{Arnoldi\_step}(Q_{r-1}, v),$$

   which, according to the Arnoldi iteration, appends these vectors (i.e., the matrix Q_{r-1}) by an orthonormal vector q_r through orthogonalizing v against $q_1, \ldots, q_{r-1}$ and also outputs the numbers $h_r := (h_{1,r-1}, \ldots, h_{r,r-1})^\top \in \mathbb{R}^r$, where $h_{\ell,r-1} := q_\ell^\top v$ for $\ell \leq r$. You can then call `Arnoldi_step(Q; v)` within `GMRES(...)`.

2. **GMRES:** Implement a function

   $$x = \texttt{GMRES(A, b, x0, tol=1e-6, maxiter=None, N=None)},$$

   which takes as arguments

   - A : a function evaluating the matrix–vector product $v \mapsto A \cdot v$ for some matrix $A \in \mathbb{R}^{n\times n}$ (not as an array!)
   - b : a vector $b \in \mathbb{R}^n$
   - x0 : an arbitrary initial guess $x^0 \in \mathbb{R}^n$
   - tol : error tolerance as float, which is set to 10^{-6} by default
   - maxiter : optional maximum number of iterations, which is set to `None` by default
   - N : optional preconditioner as a function (not as an array), for which $N(v) \approx A^{-1}v$

   and then solves the system $Ax = b$ by applying the GMRES method as presented in the lecture (see pseudocode 5 below). It shall then return

   - x : the approximation to the solution.

   The iteration shall break if the residual is tolerably small, i.e.,

   $$\|Ax^k - b\|_2 < \texttt{tol}$$

   or the maximum number of iterations `maxiter` has been reached.

3. **Test** your solver on a random invertible tridiagonal matrix

   $$A = \begin{pmatrix} * & * & 0 & \cdots & 0 \\ * & * & * & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & * & * & * \\ 0 & \cdots & 0 & * & * \end{pmatrix} \in \mathbb{R}^{n\times n}$$

   and some right-hand side b and initial guess x_0 of your choice. Choose different n and check how many iterations you need (potentially many!).
   Hint: You can generate some random diagonals using `numpy.random.rand(n)` *and then use*

   ```
   scipy.sparse.diags(diagonals, offsets=[-1,0,1]).tocsr()
   ```

   to construct a sparse CSR matrix. You can add `np.ones(n)` *to the main diagonal in order to "strengthen" the diagonal of A and thereby to get a better conditioned system. Then implement a function* `A(x)` *that outputs the matrix-vector product* `A.dot(x)`.

4. **Preconditioner:** For the same system, run your GMRES routine with a preconditioner of your choice (for example Jacobi $N : v \mapsto D^{-1}v$). Do you observe any difference in the number of iterations needed? (You may not necessarily observe a difference!)

5. **Compare** your `GMRES` solver to `scipy.linalg.solve(A_csr.toarray(),b)` for large dimension $n \geq 10^5$ and measure the time needed in each case. Also, find a SciPy implementation of GMRES and compare to yours.

INPUT: $A \in GL_n(\mathbb{R})$, $b \in \mathbb{R}^n$
OUTPUT: approximation $x_r \in K_r(A,b)$ to the exact solution $A^{-1}b$

GMRES(A, b, $x_0 = 0$, `tol = 1e-6`, `maxiter=None`, $N = I$):
$b := b - Ax_0$ //account for initial guess
$A := NA$, $b := Nb$ //account for preconditioner
//Initialization:
$q_1 := \frac{b}{\|b\|_2}$, $Q_1 := [q_1]$
$v := Aq_1$
$\tilde{q}_1 := \frac{v}{\|v\|_2}$, $\tilde{Q}_1 := [\tilde{q}_1]$, $\tilde{R}_1 = [\|v\|_2]$
for $r = 2, ..., \min(n, \texttt{maxiter})$ **do**
  //STEP 1: use Arnoldi to find column q_r by orthogonalizing v against $q_1, \dots, q_{r-1}$
  $q_r, h_{r-1} := \texttt{Arnoldi\_step}(Q_{r-1}; v)$ //we don't need h_{r-1} in this variant
  $Q_r := [Q_{r-1}, q_r]$
  $v := Aq_r$

  //STEP 2: use Arnoldi to find columns $\tilde{q}_r$, $\tilde{r}_r$ by orthogonalizing v against $\tilde{q}_1, \dots, \tilde{q}_{r-1}$
  $\tilde{q}_r, \tilde{r}_r := \texttt{Arnoldi\_step}(\widetilde{Q}_{r-1}; v)$
  $\widetilde{Q}_r := [\widetilde{Q}_{r-1}, \widetilde{q}_r]$, $\widetilde{R}_r := [\widetilde{R}_{r-1}, \widetilde{r}_r]$

  //STEP 3: solve auxiliary least squares problems to obtain coordinates
  $c_r := \texttt{solve\_triangular}(\widetilde{R}_r, \widetilde{Q}_r^\top b)$
  $x_r := Q_r c_r$
  //Attention: Evaluate the original residual here:
  if $\|N^{-1}(Ax_r - b)\|_2 < \texttt{tol}$ **then**
    break
  end
end
return $x_0 + x_r$

Algorithm 5: GMRES

Ex 112 Linear Systems, Splitting Methods, Python

Splitting Methods: relax. Richardson, relax. Jacobi, Gauß-Seidel and SOR

1. Implement a function

   ```
   x, error, numiter = iter_solve(A, b, x0, method="Jacobi", theta=.1, tol=1e-08, maxiter=50)
   ```

   which takes as arguments

   - A : a matrix $A \in \mathbb{R}^{n \times n}$
   - b : a vector $b \in \mathbb{R}^n$
   - x0 : an initial guess $x^0 \in \mathbb{R}^n$
   - `method` : optional parameter to choose between relax. Richardson, weighted Jacobi, Gauß-Seidel and SOR and which is set to `"Jacobi"` by default
   - theta : relaxation parameter θ which is set to 0.1 by default
     (note: Gauß–Seidel is SOR with `theta=1.0`)
   - tol : error tolerance as float, which is set to 10^{-8} by default
   - maxiter : maximum number of iterations, which is set to 50 by default

   and then solves the system $Ax = b$ by applying the specified iterative scheme. It shall then return

   - x : list of all iterates x^k
   - `error` : list containing all residuals $\|Ax^k - b\|_2$
   - `numiter` : number of iterations that have been performed

   The iteration shall break if the residual is tolerably small, i.e.,

   $$\|Ax^k - b\|_2 < \texttt{tol}$$

or the maximum number of iterations `maxiter` has been reached. Implement the **element-based** formulas for the Jacobi, Gauß-Seidel and SOR method.

2. **2d:** Test all methods on the following two-dimensional setting:

$$A = 4\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

What is the exact solution x^*? Play around with the parameters `x0, theta, tol` and `maxiter`. Also create the following two plots for one fixed setting:

- Plot the error $\|Ax^k - b\|_2$ for each iterate $x^k \in \mathbb{R}^2$, $k = 1, \ldots, m$, for all methods into one plot (use different colors).
- Plot the iterates $x^k \in \mathbb{R}^2$, $k = 1, \ldots, m$, for all methods into one plot (use different colors).

3. **nd:** Next, test all methods on the higher–dimensional analogue

$$A = n^2 \begin{pmatrix} 2 & -1 & 0 & \ldots & 0 \\ -1 & 2 & -1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & -1 & 2 & -1 \\ 0 & \ldots & 0 & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n \times n},$$

for different dimensions $n \in \mathbb{N}$ and data $b, x^0 \in \mathbb{R}^n$ of your choice. Play around with the parameters.

Hint: It can happen that the iterations do not converge. Use small values for θ when you use the Richardson iteration. This will assure that $\rho(I - NA) < 1$.

Ex 113

Richardson Iteration
Let $A \in \mathbb{R}^{n \times n}$ be a symmetric and positive definite matrix. Consider the (relaxed) Richardson iteration

$$x_{k+1} = (I - \theta A)x_k + \theta b$$

with (symmetric) iteration matrix $M_\theta := I - \theta A$.

1. Let λ_{max} (λ_{min}) denote the largest (smallest) eigenvalue of A. Show that the spectral radius of M_θ is given by

$$\rho(M_\theta) = \max\{|1 - \theta\lambda_{max}|,\ |1 - \theta\lambda_{min}|\}.$$

Hint: Note that A has only positive eigenvalues. Determine the spectrum of M_θ by observing that it is a scaled and shifted version of A.

2. Determine all $\theta \in \mathbb{R}$ for which the Richardson iteration converges.
Hint: Use 1. and find all $\theta \in \mathbb{R}$ for which $\rho(M_\theta) < 1$.

3. Draw the function $\theta \mapsto \rho(M_\theta)$ to determine the optimal $\hat{\theta}$, which fulfills $\rho(M_{\hat{\theta}}) \leq \rho(M_\theta)$ for all $\theta \in \mathbb{R}$.
Hint: We find $\hat{\theta} = \frac{2}{\lambda_{max} + \lambda_{min}}$.

4. Show that for the optimal relaxation parameter $\hat{\theta}$ it holds that

$$\rho(M_{\hat{\theta}}) = \frac{\mathrm{cond}(A) - 1}{\mathrm{cond}(A) + 1},$$

where $\mathrm{cond}(A) := \frac{\lambda_{max}}{\lambda_{min}}$ denotes the condition number of the symmetric matrix A. What impact does a large condition number (i.e., $\mathrm{cond}(A) \gg 1$) have on the convergence speed?

Ex 114 Linear Systems, Splitting Methods

Estimate Number of Iterations
Consider the linear system $Ax = b$ with

$$A = \begin{pmatrix} 3 & -1 \\ -1 & 3 \end{pmatrix}.$$

Assume you want to solve this system with the Richardson, Jacobi and Gauß-Seidel method, respectively, without relaxation ($\theta = 1$). Estimate the number m of iterations that are needed for each method (if convergent) to reduce the error by a factor of $\varepsilon = 10^{-6}$, i.e.,

$$\|x_k - x\|_2 \leq \varepsilon \|x_0 - x\|_2.$$

Hint: Use the error estimate from previous exercises. You can later numerically verify your results.

7 Least Squares Problems

Ex 115

Let $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. Assume you are given the least squares problem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2.$$

1. Which equation does a solution $\hat{x}$ of the above least squares problem solve?
2. Assume you are given the following data

z	-2	-1	0	1	2
y	-1	0	0	2	9

   Solve the least squares problem

$$\min_{c_0, c_1 \in \mathbb{R}} \sum_{i=1}^{5} (c_0 + c_1 z_i - y_i)^2.$$

Ex 116

Let $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. Assume you are given the least squares problem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2.$$

1. Which equation does a solution $\hat{x}$ of the above least squares problem solve? Give formula and name of the equation.
2. Assume you are given the following data

z	-3	-1	0	1	3
y	-3	-1,5	0	2,5	4

   Solve the curve fitting problem

$$\min_{c_0, c_1 \in \mathbb{R}} \sum_{i=1}^{5} (c_0 + c_1 z_i - y_i)^2,$$

   i.e., determine the minimizing parameters c_0 and c_1.

Ex 117

Let $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. Assume you are given the least squares problem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2.$$

1. Which equation does a solution $\hat{x}$ of the above least squares problem solve? Give formula and name of the equation.
2. Assume you are given the following data

z	-2	-1	0	1	2
y	3,5	2,5	1	0,5	-2,5

   Solve the curve fitting problem

$$\min_{c_0, c_1 \in \mathbb{R}} \sum_{i=1}^{5} (c_0 + c_1 z_i - y_i)^2,$$

   i.e., determine the minimizing parameters c_0 and c_1.

Ex 118

Let $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. Assume you are given the least squares problem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2.$$

1. Which equation does a solution $\hat{x}$ of the above least squares problem solve? Give formula and name of the equation.
2. Assume you are given the following data

z	-3	-1	0	1	3
y	-2,5	-1,5	0	2,5	4,5

Solve the curve fitting problem

$$\min_{c_0,c_1\in\mathbb{R}} \sum_{i=1}^{5} (c_0 + c_1 z_i - y_i)^2.$$

Ex 119

Let $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times n}$. Assume you are given the least squares problem

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|_2^2.$$

1. Which equation does a solution $\hat{x}$ of the above least squares problem solve? Give formula and name of the equation.
2. Assume you are given the following data

z	-3	-1	0	1	3
y	-2,5	-1,5	0	2,5	4,5

Solve the curve fitting problem

$$\min_{c_0,c_1\in\mathbb{R}} \sum_{i=1}^{5} (c_0 + c_1 z_i - y_i)^2.$$

Ex 120

Let $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Assume you are given the least squares problem

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|_2^2.$$

1. Which equation does a solution $\hat{x}$ of the above least squares problem solve?
2. Assume you are given the following data

z	-2	-1	0	1	2
y	3	-1	0	1	4

Solve the curve fitting problem

$$\min_{c_0,c_1\in\mathbb{R}} \sum_{i=1}^{5} (c_0 + c_1 z_i^2 - y_i)^2,$$

i.e., determine the minimizing parameters c_0 and c_1.

Ex 121

Least Squares
We are given a sample of size m of measurements $(z_i, y_i) \in \mathbb{R}^2$ for $i = 1, \ldots, m$. Determine the minimizer c_0 of the problem

$$\min_{c_0\in\mathbb{R}} \sum_{i=1}^{m} (c_0 - y_i)^2.$$

Ex 122

Minimum Norm Least Squares Solution: Example
Let $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times n}$. Assume you are given the least squares problem

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|_2^2.$$

1. Which equation does a solution $\hat{x}$ of the above least squares problem solve? Give formula and name of the equation.
2. Assume you are given the following data

z	-1	1
y	2	-1

which you want to explain by a model $f : \mathbb{R} \to \mathbb{R}$ of the form

$$f_c(z) = c_0 + c_1 z + c_2 z^2.$$

Solve the least squares problem

$$\min_{c \in \mathbb{R}^3} \sum_{i=1}^{2} (f_c(z_i) - y_i)^2$$

to determine appropriate coefficients (c_0, c_1, c_2). If there are infinitely many solutions, pick the one with minimal norm.

Hint: Characterize the solution set $\hat{S}$ and derive a formula for the norm of a solution. Then use the fact, that the minimum (or maximum) of a quadratic function $p\colon \mathbb{R} \to \mathbb{R}$, $p(s) = a_0 + a_1 s + a_2 s^2$ can be found by setting the first derivative to zero, i.e., $0 = p'(s) = a_1 + 2a_2 s$.

Ex 123 Least Squares Problems

Orthogonal Projection
Prove the following statement: Let $V \subset \mathbb{R}^m$ be a linear subspace and $b \in \mathbb{R}^m$. Then

$$\widehat{z} = \arg\min_{z \in V} \|z - b\|_2^2 \quad \Leftrightarrow \quad \widehat{z} - b \in V^\perp := \{w \in \mathbb{R}^n \colon w^\top v = 0 \ \ \forall v \in V\}.$$

Hint: You can use: For all $x, y \in \mathbb{R}^m$: $\|x + y\|_2^2 = \|x\|_2^2 + \|y\|_2^2 \ \Leftrightarrow \ x^\top y = 0$.

Ex 124 Linear Algebra, Singular Values, Python

Least Squares and *Total* Least Squares (Principal Components)

1. Use the function `numpy.random.multivariate_normal()` to create samples $(z_i, y_i) \in \mathbb{R}^2$ for $i = 1, \ldots, 100$ from a 2-dimensional multivariate normal distribution with mean $\mu := (0, 0)$ and covariance

$$\Sigma := \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}.$$

2. Implement a function or find a Scipy function to solve the least squares problem

$$\min_{c \in \mathbb{R}} \sum_{i=1}^{100} (cz_i - y_i)^2$$

and plot the model $f(z) = cz$ and the data points into one plot.

3. Now consider the 2×100 matrix $A = [v_1, \cdots, v_n]$ with columns $v_i := (z_i, y_i)^\top \in \mathbb{R}^2$ and compute an SVD $A = U\Sigma V^\top$ of it. Draw the lines through the column vectors of U into the same plot. These are the principal components that explain most of the variance.

Ex 125 Least Squares Problems, Python

Ridge Regression and the Minimum Norm Solution
Let $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. Assume you are given the regularized least squares problem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 + \tfrac{\delta}{2}\|x\|_2^2.$$

1. Which equation does the solution x_δ of the above least squares problem solve?
2. Assume you are given the following data

z	-1	1
y	2	-1

which you want to explain by a model $f : \mathbb{R} \to \mathbb{R}$ of the form

$$f_c(z) = c_1 + c_2 z + c_3 z^2.$$

Implement a program to solve the regularized least squares problem

$$\min_{c \in \mathbb{R}^3} \sum_{i=1}^{2} (f_c(z_i) - y_i)^2 + \tfrac{\delta}{2} \sum_{j=1}^{3} c_j^2$$

to determine appropriate coefficients $x_\delta = (c_1^\delta, c_2^\delta, c_3^\delta)$ for *multiple* $\delta \in (0, 1)$. Why is regularization an appropriate approach here?

3. Find a routine to compute the minimum norm least squares solution x^+ and compare it to your solutions x_δ. What do you observe for small δ?
4. Plot the measurements and the fitting polynomial corresponding to x_δ and x^+ into one figure. What do you observe for small δ?

   Hint: Use

   - `scipy.linalg.solve` to solve a linear system and set the correct flag that informs the function about the positivity of the matrix (see documentation),
   - `numpy.linspace` to create an array of multiple $\delta \in (0, 1)$,
   - the plot routines from previous exercises if you want.

Ex 126 Linear Algebra

Projections and Least Squares
Let $a, b \in \mathbb{R}^n \setminus \{0\}$ be two nonzero vectors. Consider the 1-dimensional optimization task

$$\min_{c \in \mathbb{R}} \frac{1}{2} \|ca - b\|_2^2 =: f(c),$$

where $\|x\|_2 := \left(\sum_{i=1}^n x_i^2\right)^{\frac{1}{2}}$ denotes the Euclidean norm of a vector $x \in \mathbb{R}^n$.

1. Determine the parameter $c \in \mathbb{R}$ which minimizes f.
   Hint: As in high-school, compute the derivative f' of f with respect to c and solve the equation $f'(c) = 0$.
2. Compare your results to the projection of b onto a, i.e., $\text{proj}_a(b) := \frac{a^\top b}{\|a\|_2} \frac{a}{\|a\|_2}$.

Ex 127 Least Squares Problems

Properties of $A^T A + \delta I$
Let $A \in \mathbb{R}^{m \times n}$ be any matrix.

1. Please show that $A^T A + \delta I$ is symmetric and positive definite for all $\delta > 0$.
2. Why is $A^T A + \delta I$ invertible for all $\delta > 0$?

Ex 128

---

8 Vector Spaces

Ex 129 Linear Algebra

Proof of Lemma 11.15
Let $f \in \mathrm{Hom}_{\mathbb{F}}(\mathbb{F}^n, \mathbb{F}^m)$ with representing matrix $A \in \mathbb{F}^{m \times n}$. Please show:

1. The sets $\ker(f)$ and $\mathrm{Im}(f)$ are subspaces of $\mathbb{F}^n$ and $\mathbb{F}^m$, respectively.
2. $\mathrm{rank}(A) = \dim(\mathrm{Im}(f))$.
3. $\ker(f) = \{0\} \quad \Leftrightarrow \quad f$ is injective.

Ex 130 Linear Algebra

Vector Space of Polynomials
Let $n \in \mathbb{N}$ and $P_n(\mathbb{R})$ be the set of all polynomials of degree $\leq n$ on $\mathbb{R}$, i.e., the set $P_n(\mathbb{R})$ contains all functions $p : \mathbb{R} \to \mathbb{R}$ of the form

$$p(x) = \sum_{k=0}^{n} \alpha_k x^k$$

for some $\alpha_0, \alpha_1, \ldots, \alpha_n \in \mathbb{R}$. We define a summation and scalar multiplication:

$$\begin{aligned} +&: P_n(\mathbb{R}) \times P_n(\mathbb{R}) \to P_n(\mathbb{R}), \ (p+q)(x) := p(x) + q(x), \\ \cdot&: \mathbb{R} \times P_n(\mathbb{R}) \to P_n(\mathbb{R}), \qquad (r \cdot p)(x) := r \cdot p(x). \end{aligned}$$

1. **VR axioms:** Please show that $P_n(\mathbb{R})$ together with the above defined summation and scalar multiplication forms a vector space.
2. Let $k < m \in \mathbb{N}$. Compute
$$\lim_{x \to \infty} \frac{x^k}{x^m}, \quad \text{and} \quad \lim_{x \to \infty} \frac{\sum_{k=0}^{m-1} \alpha_k x^k}{x^m}$$
for arbitrary $\alpha_0, \ldots, \alpha_m \in \mathbb{R}$.
3. **Monomials form a basis:** Please show that the set $B := \{q_0, \ldots, q_n\}$ with
$$q_k : \mathbb{R} \to \mathbb{R}, \ \ x \mapsto x^k,$$
is a basis of $P_n(\mathbb{R})$. What is the dimension of the vector space $P_n(\mathbb{R})$? *Hint: Part 2. basically provides the proof of linear independence.*
4. **Derivative as linear operator:** Show that the operator $\mathcal{D} : P_n(\mathbb{R}) \to P_{n-1}(\mathbb{R}), \ \ p \mapsto p'$, which maps a polynomial to its first derivative, is an $\mathbb{R}$-linear function.
Hint: You can use that for $p(x) = \sum_{k=0}^{n} \alpha_k x^k$ we have $Dp(x) = p'(x) = \sum_{k=0}^{n-1} \alpha_{k+1}(k+1)x^k$.
5. **Matrix representation of the derivative:** Derive the matrix representation of $\mathcal{D}$ with respect to the bases of monomials, i.e., derive the matrix $\mathcal{M}_{\{q_0,\ldots,q_{n-1}\}}^{\{q_0,\ldots,q_n\}}(\mathcal{D})$.

9 Calculus

Ex 131 Derivatives

Derivatives
Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix and $b \in \mathbb{R}^n$ a vector . Show that the function

$$f : \mathbb{R}^n \to \mathbb{R}, x \mapsto \tfrac{1}{2}x^T Ax - b^T x$$

is Frechét differentiable and determine the gradient $\nabla f(x)$ of f at a point $x \in \mathbb{R}^n$.

Hint: Compute the directional (Gâteau) derivative and use the resulting expression as a candidate for the Fréchet derivative.

Ex 132 Derivatives

Differentiable implies Continuous
Let $D \subset \mathbb{R}^n$, $x_0 \in D$ with $B_\varepsilon(x_0) \subset D$ for some $\varepsilon > 0$. Let $f\colon D \to \mathbb{R}$ be (Frechét-) differentiable at x_0. Show that f is continuous at x_0.

Ex 133

Heron's Method as Newton's Method
Consider the function $f\colon \mathbb{R} \to \mathbb{R},\ f(x) := x^2 - a$ for some nonnegative number $a \geq 0$. Apply Newton's method to the nonlinear system $f(x) = 0$ (root finding problem). Compare the resulting iterative scheme to Heron's algorithm from earlier sheets.

Ex 134 Linear Systems, Splitting Methods, Python

Appetizer: Gradient, Steepest Descent and Conjugate Gradient Method

Background: Let $A \in \mathbb{R}^{n\times n}$ be symmetric and positive definite (spd). Then A is in particular invertible, so that the linear system $Ax = b$ has a unique solution $x^* \in \mathbb{R}^n$ for all $b \in \mathbb{R}^n$. Let us relate this linear system to an optimization problem. For this purpose we define for a fixed spd matrix A and fixed right-hand side b the function

$$f := f_{A,b} : \mathbb{R}^n \to \mathbb{R}, \quad x \mapsto \tfrac{1}{2}x^T Ax - b^T x.$$

Then one can show the equivalence

$$Ax^* = b \quad \iff \quad x^* = \arg\min_{x\in\mathbb{R}^n} f(x).$$

In words, x^* solves the linear system on the left-hand side if and only if x^* is the unique minimizer of the functional f. In fact, you will learn in the next semester that the condition $Ax^* = b$ is the necessary first-order optimality condition:

$$0 = \nabla f(x) = Ax - b.$$

Due to the convexity of f this condition is also sufficient. Consequently, solving linear systems which involve spd matrices is equivalent to solving the associated optimization problem above, i.e., minimizing the function $f(x) = \frac{1}{2}x^T Ax - b^T x$. Thus, in this context iterative methods for linear systems, such as the Richardson iteration, can also be interpreted as optimization algorithms. Let us consider the (relaxed) Richardson iteration for $Ax = b$, i.e., $x_{k+1} = (I - \theta A)x_k + \theta b$. After some minor manipulations and making use of $\nabla f(x_k) = Ax_k - b$ we arrive at the equivalent formulation

$$x_{k+1} = x_k - \theta \nabla f(x_k).$$

The latter is what is called a gradient method. A step from x_k into (an appropriately scaled) direction of the gradient $\nabla f(x_k)$ yields a decrease in the objective function f , i.e., $f(x_{k+1}) \leq f(x_k)$). Along the Richardson aka Gradient method the scaling (also called step size) θ is fixed (in a machine learning context the step size θ is called the *learning rate*). However, one could also choose a different θ_k in each iteration step. This gives the more general version

$$x_{k+1} = x_k - \theta_k \nabla f(x_k). \tag{1}$$

The well known method of *steepest descent* is given by choosing

$$\theta_k = \frac{r_k^\top r_k}{r_k^\top A r_k}, \tag{2}$$

where $r_k := Ax_k - b$ is the k-th residual. This choice can be shown to be optimal in terms of convergence speed. Even more general, one can think of using a different preconditioner N_k in each iteration step

$$x_{k+1} = x_k - N_k \nabla f(x_k),$$

i.e., representing the gradient with respect to another inner product. This will later correspond to Newton-type optimization algorithms.

Task
Consider the following setting:

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 10 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad x_0 = \begin{bmatrix} 4 \\ 1.4 \end{bmatrix}.$$

Convince yourself that A is spd. Determine the minimal and maximal eigenvalue of A, i.e., λ_{min} and λ_{max}, respectively. What is the solution to $Ax = b$? Now extend your code (in particular `iter_solve()`) from previous sheets:

1. Implement the **steepest descent method** (1) by choosing the stepsize θ_k from (2) in each iteration step.
2. Find a way to apply the **conjugate gradient method** to solve a system $Ax = b$, where A is spd.
   Hint: You can either implement it on your own or find a SciPy routine (for the latter: you can collect all iterates x_k by using the callback interface).
3. **Test:** Solve the above problem with the following routines:
   a) Richardson method with $\theta = \frac{2}{\lambda_{\text{max}}}$
   b) Richardson method with $\theta = 0.9 \cdot \frac{2}{\lambda_{\text{max}}}$
   c) Richardson method with optimal $\theta = \frac{2}{\lambda_{\text{min}} + \lambda_{\text{max}}}$
   d) Steepest descent method
   e) conjugate gradient method

   Generate the following two plots:
   1. Plot the iterates x_k for all the runs into the same 2d plot (use different colors).
   2. Plot the function values $f(x_k) = \frac{1}{2} x_k^T A x_k - b^T x_k$ for each iterate and all runs into a second plot (use different colors).

10 Small Tour into Imaging

Ex 135 Linear Algebra, Image Processing

Discrete Convolution, Filters and Kronecker Product

1. *Definition: Let $f\colon \mathbb{Z} \to \mathbb{R}$ and $g\colon \mathbb{Z} \to \mathbb{R}$. Then the discrete convolution $*\colon \mathbb{R}^{\mathbb{Z}} \times \mathbb{R}^{\mathbb{Z}} \to \mathbb{R}^{\mathbb{Z}}$ is defined by*

$$(f * g)(k) = \sum_{j \in \mathbb{Z}} f(j)g(k-j), \qquad k \in \mathbb{Z}.$$

Consider

$$f(i) := \begin{cases} 1 & : i \in \{0,1,2\} \\ 0 & : else \end{cases}, \qquad g(i) := \begin{cases} \frac{1}{3} & : i \in \{-1,0,1\} \\ 0 & : else \end{cases},$$

and compute the discrete convolution $f * g$.
Remark: You can think of f as a zero-padded finite signal and of g as a centered filter. Also, most values of $f * g$ are zero.

2. Consider f and g from above. Define a matrix $G \in \mathbb{R}^{3\times 3}$, so that

$$G \cdot \begin{pmatrix} f(0) \\ f(1) \\ f(2) \end{pmatrix} = \begin{pmatrix} (f*g)(0) \\ (f*g)(1) \\ (f*g)(2) \end{pmatrix}$$

Do you observe a structure in G?
Remark: The matrix G is the filter matrix for the average filter $\frac{1}{3}[1\ 1\ 1]$.

3. Now consider the 2d data

$$F := \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

and the matrix G from above. Then compute the product

$$GFG^\top.$$

Remark: By computing this product we apply the 1d filter to each dimension of F, e.g., first column-wise and then row-wise: $(G(GF)^\top)^\top$. Thereby we obtain a 2d filter from the 1d filter.

4. *Definition: The Kronecker product $\otimes\colon \mathbb{R}^{m\times n} \times \mathbb{R}^{M\times N} \to \mathbb{R}^{mM\times nN}$ is defined by*

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}.$$

Also let vec$\colon \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ define the row-major vectorization (here start indexing at 0)

$$\text{vec}(F)(k) := F(\lfloor \tfrac{k}{n} \rfloor, k \mod n), \quad 0 \le k < mn.$$

For G and F from above derive

$$\text{vec}(GFG^\top) \quad \text{and} \quad (G \otimes G)\text{vec}(F).$$

Remark: These two vectors should be the same. The right one defines the application of the 2d filter as a matrix–vector product.

5. *Definition: Let $F\colon \mathbb{Z}^2 \to \mathbb{R}$ and $W\colon \mathbb{Z}^2 \to \mathbb{R}$. Then the two-dimensional discrete convolution $*\colon \mathbb{R}^{\mathbb{Z}^2} \times \mathbb{R}^{\mathbb{Z}^2} \to \mathbb{R}^{\mathbb{Z}^2}$ is analogously defined by*

$$(F * W)(k,\ell) = \sum_{(i,j)\in\mathbb{Z}^2} F(i,j)W(k-i,\ell-j), \quad (k,\ell) \in \mathbb{Z}^2.$$

Consider

$$F(i,j) := \begin{cases} 1 & : (i,j) \in \{(1,0),(1,1),(1,2),(2,1)\} \\ 0 & : else \end{cases}, \qquad W(i,j) := \begin{cases} \frac{1}{9} & : -1 \le i,j \le 1 \\ 0 & : else \end{cases},$$

and compute the discrete convolution $F * W$. Compare the result to $GFG^\top$ from above.
Remark: This F is the zero-padded version of the 2d signal F from above and W is the 2d average filter.

6. Just for fun: Install the free software https://www.gimp.org/downloads/, open an image of your choice and check out the menu point "Filter". In particular, you can apply your own filters under "Filter > Generic > Convolution Matrix" (https://docs.gimp.org/2.8/en/plug-in-convmatrix.html).

Ex 136 Least Squares Problems, Python

More General Regularization Terms and Image Inpainting

Background:
Based on the idea of Tikhonov regularization we can use more general regularization terms by transforming the vector x with some matrix $G \in \mathbb{R}^{m\times n}$. Specifically, let us consider the more general regularized problem

$$\min_{x\in\mathbb{R}^n} \|Ax-b\|_2^2 + \frac{\delta}{2}\|Gx\|_2^2, \quad \delta > 0 \text{ small.} \tag{1}$$

The corresponding "regularized" normal equation then reads as

$$(A^TA + \delta G^\top G)x = A^Tb. \tag{2}$$

Observe that for $G = I$ we obtain the standard Tikhonov (or L^2-) regularization. We easily see that if $G^\top G$ is positive definite, then so is $A^TA + \delta G^\top G$ for positive δ, so that (2) is uniquely solvable.

We will apply this framework to the problem of image inpainting. Therefore assume you are given the deteriorated image b, which is obtained from the unknown original image x through the following masking operation

$$b_i = \begin{cases} x_i & i \in \texttt{indices}, \\ 0 & \text{else}, \end{cases} \tag{3}$$

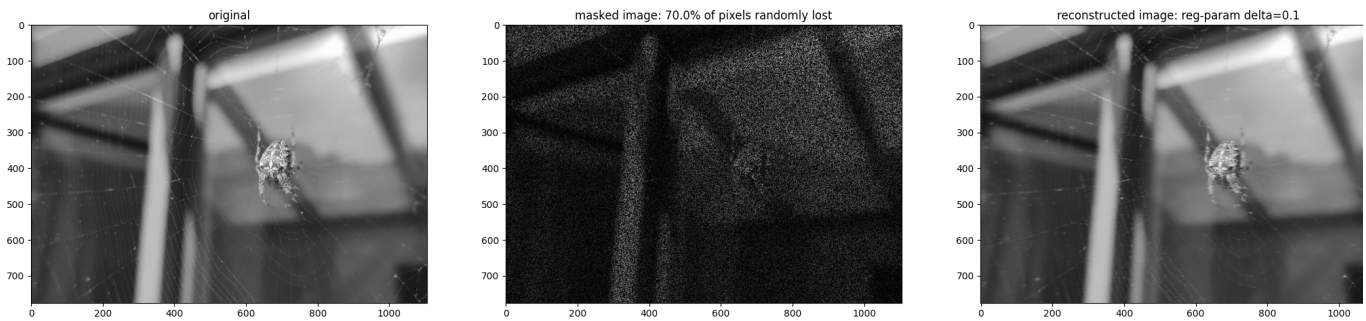
where `indices` is a list of random pixels. In words, the pixels in `indices` survived, the rest is set to zero and therefore lost. We want to recover those lost pixels. Note that the images are considered being flattened and thus vectors, so that the masking operation (3) can be written as a matrix-vector product

$$b = Ax$$

for some quadratic matrix A. You will see below that A is not of full rank, so that we cannot simply solve this equation. Instead, we will seek for solutions of the regularized least squares problem (1) by solving the linear equation (2).
For this purpose will stick to a particular G which is related to what is called Sobolev (or H^1-) regularization. Specifically we consider the 1-d finite difference quotient

$$G = \begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix} \in \mathbb{R}^{(n+1)\times n}, \tag{4}$$

which has 1 on the main diagonal, -1 on the first lower off-diagonal, 0 everywhere else and is of dimension $(n+1)\times n$. Then, given the measured image b, the masking operator A (we assume that we know which pixels are original) and the regularization G we reconstruct the unknown image x by solving equation (2). See Figure below for an example experiment.



Example experiment.

Step by Step
First note that due to the high dimensional image data you need to work with sparse matrices (`scipy.sparse`); specifically you should work with the CSR format.

1. **Choose original image:** Choose an image and load it in gray-scale as $H \times W$ `numpy.ndarray` using the code snippet:

```
def load_image_as_gray(path_to_image):
    import matplotlib
    img = matplotlib.image.imread(path_to_image)
    # ITU-R 601-2 luma transform (rgb to gray)
    img = np.dot(img, [0.2989, 0.5870, 0.1140])
    return img
```

2. **Masking:** Write a function

   `b, indices = masking(img, percentage)`,

   which takes as input an image `img` as $H \times W$ `numpy.ndarray` and a number `percentage` $\in (0, 1)$ which indicates the percentage of pixels that are randomly kept. It shall return the masked image b as an $n := (H \cdot W)$-dimensional vector and the list `indices` $\subset \{0, \dots, n-1\}$ indicating which pixels are original.
   - Ultimately, the image needs to become a vector. For this purpose you can use for example the method `.ravel()`. No matter which method/function you choose, be aware of how the vector is flattened (standard is often: row-major/C-style order).
   - To generate a random set of indices based on the parameter `percentage` have a look at the function `numpy.random.choice`.

3. **Solving:** Write a function

   `reconImg = inpainting(b, indices, delta, G)`,

   which expects a deteriorated image b as vector of length n, a list `indices` $\subset \{0, \dots, n-1\}$ of length $\leq n$ indicating which pixels are original, a regularization parameter `delta` > 0 and a matrix `G` $\in \mathbb{R}^{m \times n}$. It then solves (2) and returns the solution as an n-dimensional vector `reconImg`.
   - The sparse $(n \times n)$ masking matrix A is zero everywhere except for $a_{ii} = 1$ for $i \in$`indices`. For example, you can easily implement this matrix with `sparse.coo_matrix` and then transform it to CSR format via its method `.tocsr()`.
   - You can implement the sparse matrix G from (4) with the help of the function `scipy.sparse.eye`.
   - You can then solve the system (2) with `scipy.sparse.linalg.spsolve`.

4. **Analysis:**
   - Play around with different choices for the parameters `delta` and `percentage`.
   - You can plot your images (original, masked, reconstructed) with

     `matplotlib.pyplot.imshow((...).reshape(H,W), cmap='gray')`.

   - Bonus: Try to recover the image with standard Tikhonov regularization, i.e., $G = I$. Do you have an idea why this does not work here?
   - Bonus: With this choice of G, you get flattening artifacts in one dimension (horizontal or vertical depending on your flattening approach) and at the boundary of the reconstructed image. Do you have an idea why?

5. **Bonus:** Instead of removing randomized set of pixels, remove a patch of the image. Can you reconstruct it?

Ex 137 Image Processing, Least Squares Problems, Python

Image Inpainting

1. Download the file `noisy_image.npy` (link "data" next to the link "Sheet 11") and use `numpy.load()` to import it into your Python script. The shape is (823, 1238). This image is obtained from the original image by randomly setting 99% (!) of the pixel values to zero (=black).

2. Apply the techniques from the lecture to recover the original image:
   a) Estimate the applied masking matrix A from the given noisy image.
   b) Solve a regularized least squares problem. Note that the solving step may take a few seconds due to the high–dimensional data.

   (Can you identify the objects on the image?)