

# Introduction to Numerical Linear Algebra

Dr. Christian Vollmann

Winter Term 2022/2023

At Trier University variants of this course serve the modules:

- Elements of Mathematics
- Elemente der Linearen Algebra
- Numerical Methods for Geoscientists



The content of this work is licensed under a  
[Creative Commons "Attribution-ShareAlike 4.0  
International" License.](#)

## A Short Note on Programming

...and specifically on Python.



# Programming Languages in Numerical Mathematics (selection)

- **Fortran** (FORmula TRANslation) (1957):
  - proprietary (e.g. from IBM) and free compilers
  - intended for numerical calculations (matrix and vector operations)
  - extensive libraries
  - LAPACK (**L**inear **A**lgebra **P**ackage) standard library for numerical linear algebra
- **C** (1972)/ **C++** (1985):
  - universal programming language
  - Standard libraries for numerics: Armadillo, LAPACK++ (based on LAPACK)
- **MATLAB** (MATrrix LABoratory) (1984):
  - proprietary software from MathWorks
  - designed for numerical mathematics (matrix and vector operations)
- **Mathematica** (1988):
  - proprietary software from Wolfram Research
  - visualization of 2d/3d objects
  - symbolic processing of equations
  - see also: <https://www.wolframalpha.com/>
- **Python** (1990): open source
  - universal programming language (several application areas)
  - for numerical calculations: SciPy (2001), NumPy (1995,2006), matplotlib (2003).
- **Julia** (2012): open source
  - developed mainly for scientific computing
  - syntax looks like MATLAB
  - execution speed is in the range of C and Fortran

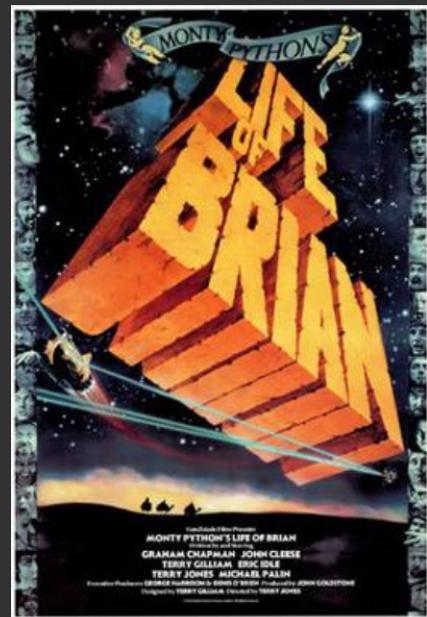
→ All programming related parts of this lecture will be presented and implemented using **Python 3**

## Why Python?

- universal, multi-purpose programming language
  - many packages for scientific computing, web development, ...
- open source and free (Python Software Foundation License (PSFL), OSI/FSF approved)
- multi-platform (runs on all operating systems)
- design philosophy: easy syntax, readable code (almost looks like pseudocode)
- Also see: <https://www.youtube.com/watch?v=M0vBoBqqjr0>

## Background

- developed in 1990 by Guido van Rossum (Netherlands)
  - name is homage to Monty Python
- interpreter (scripting) programming language  
(≠ compiled language such as C or Fortran)
- used by: Google Mail, Google Maps, YouTube, Dropbox, sphinx, and many more
- for scientific computing we use from the Scipy Stack:
  - **NumPy** (1995,2006)
  - **SciPy** (2001)
  - **matplotlib** (2003)



## Programming Workflow

### CLI [ex:demo]

- Any text editor can be used (emacs, vi, vim, nano, geany, gedit,...)
  - many editors provide syntax highlighting
- Install Python and then interpret the source code

### IDE [ex:demo]

- For software development it is often more convenient to use an **integrated development environment (IDE)**
- Specifically for Python:  PyCharm<sup>1</sup>,  Spyder

---

<sup>1</sup>sign up to jetbrains with your university account and you can get the PyCharm professional edition!

For exercise submission/presentation:



### Jupyter-Notebook [ex:demo]

- open source, *web based* interactive environment  
→ thus multi-platform
- developed by **Project Jupyter** (NPO)  
→ name refers to: **Julia, Python, R**
- the whole process can be documented:  
Coding → Documentation → Run → Communication and Presentation
- in fact, a jupyter notebook contains all the input **and** output of an interactive session plus additional text  
→ complete record!
- client VS server
  - client (local lightweight machine): browser-based workflow
  - server (remote, number cruncher): does the actual computation

## Get Started



- We recommend to download the distribution *Anaconda*:

<https://www.anaconda.com/distribution/>

→ available for Linux, Windows, and MacOS

- Comes along with:
  - graphical user interface (*Anaconda Navigator*)
  - Spyder, Jupyter Notebook, RStudio (IDE for R)
  - installs all important packages (NumPy, SciPy, matplotlib, TensorFlow, scikit-learn, \$\\dots\$)
  - package manager (*Conda*) (standard is *pip*)

## Tutorials

- Scientific computing with Python:  
<https://scipy-lectures.org/>
- Official Online-Documentation:  
<https://docs.python.org/3/>
- Official Python Tutorial:  
<https://docs.python.org/3/tutorial/index.html>
- Quickstart to Jupyter Notebook:  
<https://jupyter.readthedocs.io/en/latest/content-quickstart.html>

## Final remark:

- For Software development I would always go with an IDE due to the many additional tools: debugging, variable explorer, version control, file manager, etc.
- However, Jupyter Notebooks are very well suited for presentations and thus teaching. In particular for mandatory submissions, since the tutor can see your output, even if the program does not run on his/her machine (for whatever reasons).

# A Short Introduction to the Topic

## From the Module Handbook:

*"After completing the module, the students know the mathematical foundations in the areas of linear algebra and **numerical mathematics**. As part of the course, they acquire or deepen knowledge in the programming language **Python**."*

## Numerical mathematics?

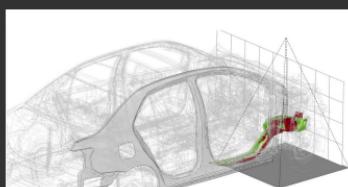
*From Wikipedia: Field of mathematics that deals with the construction and analysis of algorithms to approximately (but accurately) compute solutions to (hard) continuous problems – typically using computers.*

## Why is this important?

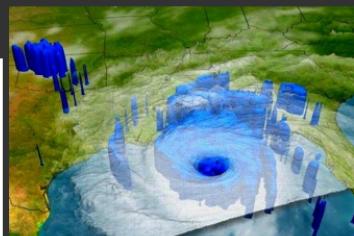
- most (all?) **application**-oriented problems cannot be solved exactly → **hard** problems



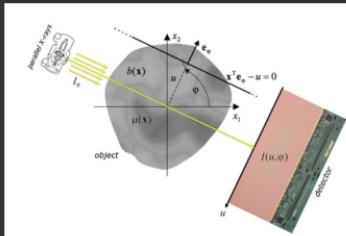
trajectories of objects in  
space  
2020SO



car crash simulation



weather prediction



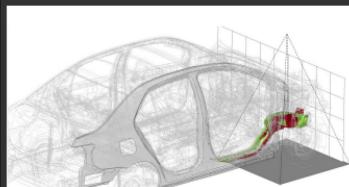
CT Scan

## Relation to Data Science?

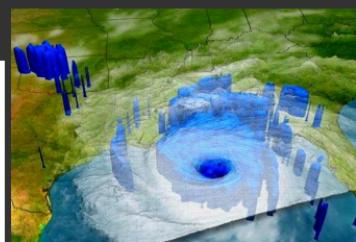
- due to the high amount of data available **data-driven** models are more important than ever
- data can be considered as a **mathematical object** (e.g., as a matrix/vector)
- with **numerical algorithms** we can manipulate data:
  - solve systems involving the data (fitting data, prediction,...)
  - extract the most important features (singular values, PCA, data compression,...)
  - calibrate models against data (machine learning, neural networks,...)



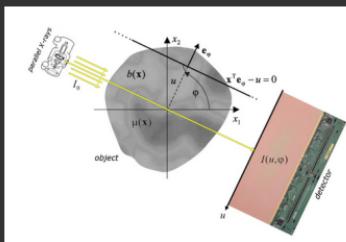
trajectories of objects in  
space  
2020SO



car crash simulation



weather prediction



CT Scan

## Preview

Let us assume we have  $m$  data points

$$(z_i, y_i), \quad i = 1, \dots, m,$$

where

- $z_i$  are  $n$ -dimensional vectors of *explanatory features*
- $y_i$  are  $k$ -dimensional vectors representing the *response/prediction/classification*

→ *The term “vector” already indicates that Linear Algebra comes naturally into the game.*

## Examples

- You ask  $m$  persons about

$$z_i = (\text{age, sex, weight, height, years of experience}) \quad (n = 5 \text{ dimensional vector})$$

$$y_i = \text{salary} \quad (k = 1 \text{ dimensional vector})$$

- Consider  $m$  years where

$$z_i = \text{year} \quad (n = 1 \text{ dimensional vector})$$

$$y_i = \text{global mean temperature} \quad (k = 1 \text{ dimensional vector})$$

- Consider  $m$  images that you want to classify

$$z_i = (p_{lj})_{lj} \quad (\text{image stored as matrix/vector})$$

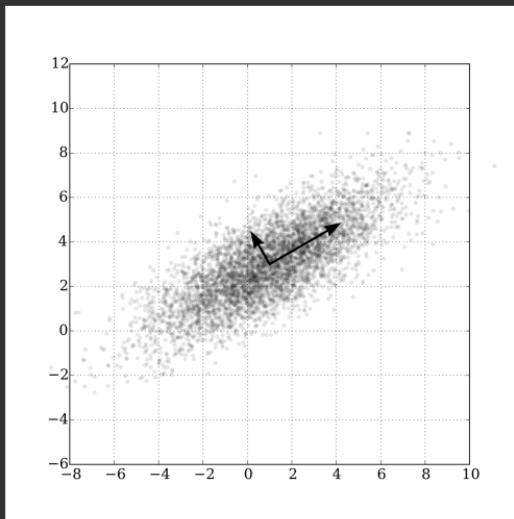
$$y_i = (\text{dog, cat, elephant}) \quad (k = 3 \text{ dimensional vector})$$

## Dealing solely with the features $z$

Applications of the **Singular Value Decomposition** are:

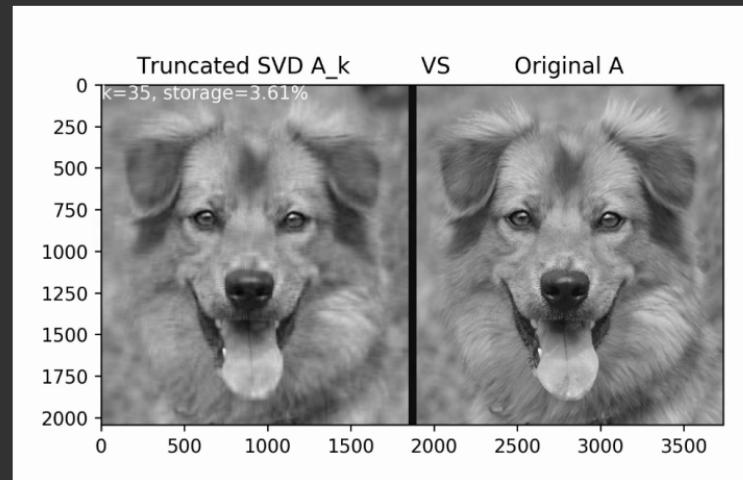
### Principal Component Analysis (PCA)

→ Aim: dimension reduction



### Data compression

→ Aim: compression without dimension reduction



## Relating features $z$ to response $y$

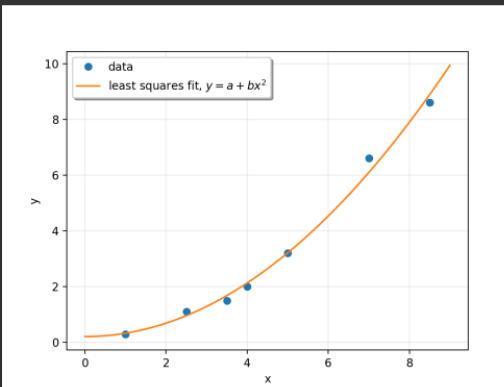
One central goal in many scientific fields is to find a **model**  $f_x$  depending on some parameters  $x = (x_i)_i$ , which “best” explains the relation between  $z_i$  and  $y_i$  in the sense that

$$f_x(z_i) \approx y_i, \quad \text{for all } i = 1, \dots, m$$

- **The task:** Find those parameters  $x$  for which the “distance” between our prediction  $f_x(z_i)$  and the measured response  $y_i$  is “as small as possible”.
- Math gives us the tools to rigorously define this task, to analyze it systematically and to provide numerical solutions!

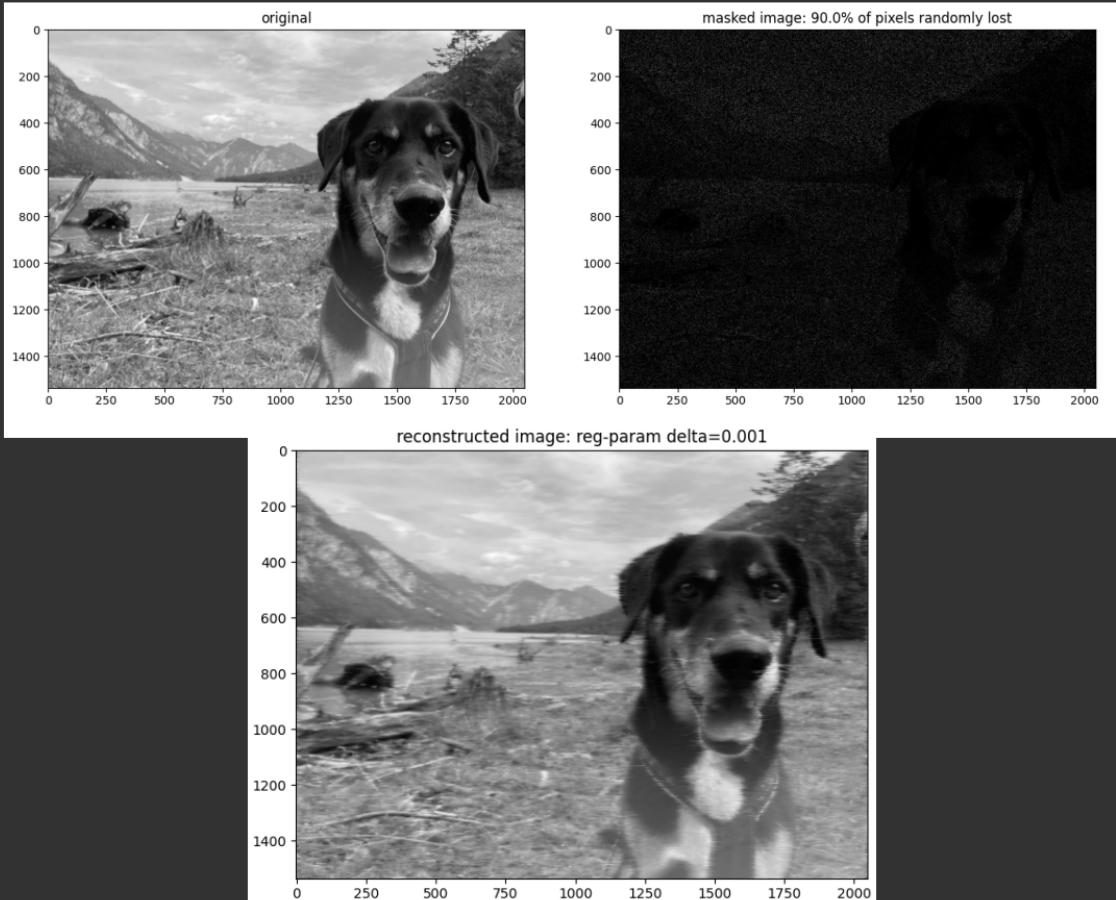
## Curve Fitting

$$f_x(z) := x_0 + x_1 z + x_2 z^2$$



## Simple image inpainting

$$\min_x \|Ax - b\|^2 + R(x)$$



## Take away message

Math provides a rigorous framework of abstract structures (Definition) within which their properties and relations (Theorem) can be systematically analyzed (Proof).

The same mathematical concept can be exploited in multiple applications; see, e.g., SVD above.

Given a real-world problem the user has to provide an interface (a mathematical model) to these concepts to make use of the mathematical theory.

## **Math Basics**

# 0 Mathematical Basics

In this first section we will provide mathematical notations and concepts which are fundamental and crucial for the further presentation.

## 0.1 Statements

By a **statement** we mean a linguistic or mental construction that is either true or false.

### Example 0.1

- “4 is an even number.” is a true statement.
- “Bananas have conic shape.” is a false statement.
- “In the night, it is colder than outside.” is not a statement.
- “There are infinitely many stars.” is a statement, which can be true or false.

## Relations and Operations

$\neg A$ :      *A is false (negation)*

$A \Rightarrow B$ : *from A follows B; if A is true, then also B is true (implication)*

*we say: A is sufficient for B, B is necessary for A*

$A \Leftrightarrow B$ : *A is true, if and only if B is true. (equivalence)*

Note that the following two statements are equivalent

$$A \Rightarrow B$$

$$\neg B \Rightarrow \neg A$$

## 0.2 Sets

**Definition 0.2 (Set)** According to Cantor a *set* is a well-defined collection of distinct objects, considered as an object in its own right. The objects that make up a set (also known as the set's *elements*) can be anything: numbers, people, letters of the alphabet, other sets, and so on.

**Notation:** curly brackets  $\{\}$

**Example 0.3**

**Definition 0.4 (Cardinality)** If a set  $M$  is *finite* (i.e., it only contains finitely many elements), then we denote by  $|M|$  the number of elements contained in  $M$  and call it *cardinality of  $M$* .

## Set relations and further definitions

$a \in M$  (or  $M \ni a$ ):  $a$  is element of  $M$ ;  $M$  contains  $a$

$a \notin M$  (or  $M \not\ni a$ ):  $a$  is not element of  $M$ ;  $M$  does not contain  $a$

$M = N$ :  $M$  contains the same elements as  $N$

$M \neq N$ :  $M$  does not contain the same elements as  $N$

$M \subset N$  (or  $M \subseteq N$ ):  $M$  is subset of  $N$ , i.e., each element of  $M$  is also an element of  $N$ ; equality of sets is permitted.

$N \supset M$  (or  $N \supseteq M$ ):  $N$  is superset of  $M$ ; analogously

$M \subsetneq N$ :  $M$  is strict subset of  $N$ ;  $M \neq N$

$\emptyset = \{\}$ : empty set

**Remark:** Very useful in practice to show that two sets are equal:

$$M = N \iff M \subset N \text{ and } N \subset M$$



$\mathcal{P}(M)$  **power set of  $M$**  defined by

$\mathcal{P}(M) := 2^M := \{N : N \subset M\}$  (set of all subsets of  $M$ )

We find  $|\mathcal{P}(M)| = 2^{|M|}$

## Summary: Set relations and further definitions

$a \in M$ (or $M \ni a$ ):	$a$ is element of $M$ ; $M$ contains $a$
$a \notin M$ (or $M \not\ni a$ ):	$a$ is not element of $M$ ; $M$ does not contain $a$
$M = N$ :	$M$ contains the same elements as $N$
$M \neq N$ :	$M$ does not contain the same elements as $N$
$M \subset N$ (or $M \subseteq N$ ):	$M$ is subset of $N$ , i.e., each element of $M$ is also an element of $N$ ; equality of sets is permitted.
$N \supset M$ (or $N \supseteq M$ ):	$N$ is superset of $M$ ; analogously
$M \subsetneq N$ :	$M$ is strict subset of $N$ ; $M \neq N$
$\emptyset = \{\}$ :	empty set
$M \times N$ :	<b>Cartesian product</b> defined by $M \times N := \{(m, n) : m \in M, n \in N\}$
	$M^n := M \times \dots \times M$ ( $n$ times)
$\mathcal{P}(M)$	<b>power set of <math>M</math></b> defined by
	$\mathcal{P}(M) := 2^M := \{N : N \subset M\}$ (set of all subsets of $M$ )
	We find $ \mathcal{P}(M)  = 2^{ M }$

## Set operations

$$M \cup N := \{a : a \in M \text{ or } a \in N\} \quad (\text{union})$$

$$M \cap N := \{a : a \in M \text{ and } a \in N\} \quad (\text{intersection})$$

$$M \setminus N := \{a : a \in M \text{ and } a \notin N\} \quad (\text{difference})$$

If  $N \subset M$

$$N^c := \overline{N} := M \setminus N \quad (\text{complement of } N \text{ with respect to } M)$$

$M, N$  are called **disjoint**, if  $M \cap N = \emptyset$ .

### Example 0.5

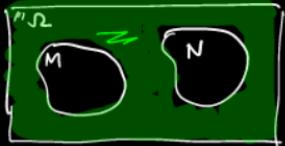
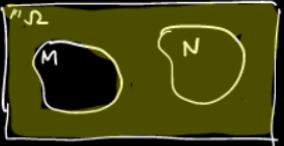
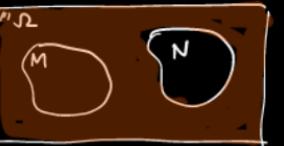
For combinations of those set operations we have the following result:

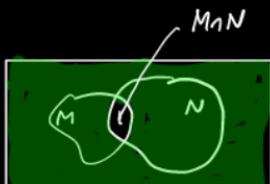
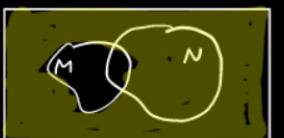
**Lemma 0.6 (De Morgan's laws)** Let  $\Omega$  be a set and  $M, N \subset \Omega$ . Then we find

- i)  $(M \cup N)^c = M^c \cap N^c$ ,
- ii)  $(M \cap N)^c = M^c \cup N^c$ .

Here the complements are taken with respect to  $\Omega$ .

*Proof.* Exercise.

i)  =  ∩ 

ii)  =  ∪ 

## 0.3 Functions

**Definition 0.7 (function)** Let  $M, N$  be two sets. A **function** or **mapping**  $f$  from  $M$  to  $N$  (notation:  $f: M \rightarrow N$ ) is determined by

- its **domain**  $M$ ,
- its **codomain**  $N$ ,
- and a **rule**,

that uniquely assigns to each element  $a \in M$  an  $b := f(a) \in N$  (notation:  $a \mapsto f(a)$ ).

Two functions  $f_1 : M_1 \rightarrow N_1$  and  $f_2 : M_2 \rightarrow N_2$  are called **equal** (abbr.  $f_1 \equiv f_2$ ) (identical), if  $M_1 = M_2$ ,  $N_1 = N_2$  and  $f_1(a) = f_2(a)$  for all  $a \in M_1$  (i.e., equal, domain, codomain and rule).

### Example 0.8

We introduce function related sets:

**Definition 0.9 (Image, preimage, graph)** Let  $A \subset M$  and  $B \subset N$ , then

- i) the set  $f(A) = \{f(a) : a \in A\} \subset N$  is called **image set** of  $A$  (under  $f$ ),
- ii) the set  $f^{-1}(B) = \{a \in M : f(a) \in B\} \subset M$  is called **preimage** of  $B$  (under  $f$ ),
- iii) the set  $\text{graph}(f) := \{(a, f(a)) : a \in M\} \subset M \times N$  is called the **graph** of  $f$ .

→ **Attention:** Here,  $f^{-1}$  is not the inverse function (see below).

[abstract picture with with image, preimage and graph]

Important properties of functions:

**Definition 0.10 (Injective, surjective, bijective)** A function  $f : M \rightarrow N$  is called

- i) **injective (one-to-one)**, if  $f(a) \neq f(\tilde{a})$  for all  $a, \tilde{a} \in M$  with  $a \neq \tilde{a}$ ;
- ii) **surjective (onto)**, if for all  $b \in N$  there exists an  $a \in M$  with  $f(a) = b$  (or equivalently  $f(M) = N$ );
- iii) **bijective**, if  $f$  is injective as well as surjective relation.

We can invert bijective functions:

**Definition 0.11 (Inverse function)** Let  $f : M \rightarrow N$  be a bijective (invertible) function. Then there exists a (unique) function  $f^{-1} : N \rightarrow M$ , the so-called **inverse of  $f$** , such that

$$f(a) = b \iff f^{-1}(b) = a.$$

**Example 0.12**

We can concatenated two or more functions:

**Definition 0.13 (Composition)** Let  $f: M \rightarrow N$  and  $g: N \rightarrow P$  be functions, then we call the function

$$g \circ f: M \rightarrow P, a \mapsto g(f(a))$$

*composition of  $f$  and  $g$ .*

[abstract picture for composition of the functions

A function with “no effect”:

**Definition 0.14 (Identity function)** Let  $M$  be a set. Then the function

$$id := id_M: M \rightarrow M, a \mapsto a$$

is called the *identity function on  $M$* .

## 0.4 Numbers

The notion of **number** has been extended over the centuries, here we do not go in detail through the axiomatic construction but just point out some properties that are useful in the remaining.

Here is an overview:

$\mathbb{N}$	<b>Natural</b>	$\{1, 2, 3, 4, 5, \dots\}$	counting objects <b>order relation:</b> $m \leq n$ $m + n, m \cdot n$ proof concept of <b>induction</b>
$\mathbb{Z}$	<b>Integer</b>	$\{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$	adding zero and negative numbers (borrowing money,...) $(\mathbb{Z}, +)$ <b>ordered, commutative group</b>
$\mathbb{Q}$	<b>Rational</b>	$\left\{ \frac{p}{q} : p, q \in \mathbb{Z}, q \neq 0 \right\}$	adding fractions of objects (one half of a cake) $(\mathbb{Q}, +, \cdot)$ <b>ordered field</b>
$\mathbb{R}$	<b>Real</b>	$\mathbb{Q} \cup \{\text{limits of sequences in } \mathbb{Q}\}$	adding square roots ( $\sqrt{2}, \sqrt{5}, \dots$ ), $\pi, \dots$ $(\mathbb{R}, +, \cdot)$ <b>ordered and complete field</b>
$\mathbb{C}$	<b>Complex</b>	$\{a + ib : a, b \in \mathbb{R}\}, i := \sqrt{-1}$	adding e.g. square root of negative numbers $\mathbb{R} \times \mathbb{R}$ with a special multiplication $(\mathbb{C}, +, \cdot)$ <b>complete field</b> (not ordered)

We have

$$\mathbb{N} \subsetneq \mathbb{Z} \subsetneq \mathbb{Q} \subsetneq \mathbb{R} \subsetneq \mathbb{C}$$

### 0.4.1 Complex Numbers $\mathbb{C}$

$\mathbb{C} = \text{"}\mathbb{R} \times \mathbb{R} \text{ with a special multiplication"}$

- extension: e.g., imaginary unit  $i := \sqrt{-1}, \sqrt{-3}, \dots$
- in real life: electricity and roots of polynomials (e.g., which do not touch the  $x$ -axis)

**Definition 0.15 (Complex numbers  $\mathbb{C}$ )** We define the field of complex numbers  $(\mathbb{C}, +, \cdot)$  by  $\mathbb{C} := \mathbb{R} \times \mathbb{R}$  with the binary operations

$$\begin{aligned}+ &: (a_1, b_1) + (a_2, b_2) := (a_1 + a_2, b_1 + b_2), \\ \cdot &: (a_1, b_1) \cdot (a_2, b_2) := (a_1 a_2 - b_1 b_2, a_1 b_2 + a_2 b_1).\end{aligned}$$

Note that  $\mathbb{R}$  itself is identified with  $\mathbb{R} \times \{0\} \subset \mathbb{C}$ .

#### Remarks

- the product in  $\mathbb{C}$  is all the magic
- $(\mathbb{C}, +, \cdot)$  is a (*complete*) **field**
- as a 2-dimensional object,  $\mathbb{C}$  does **not** possess an order relation

In order to alleviate the memorizing of the product definition, it is customary to use the so-called **imaginary unit**  $i := \sqrt{-1}$  and perform computations as if it would be a real number:

For  $z = (a_1, b_1)$ ,  $w = (a_2, b_2) \in \mathbb{C}$  we write

$$z = a_1 + ib_1 \text{ and } w = a_2 + ib_2.$$

Then the product naturally computes as

$$z \cdot w = (a_1 + ib_1) \cdot (a_2 + ib_2) = a_1a_2 + ib_1a_2 + ia_1b_2 + \underbrace{i^2}_{-1} b_1b_2 = (a_1a_2 - b_1b_2) + i(a_1b_2 + a_2b_1).$$

### Example 0.16

[note on real and imaginary part, complex conjugate]

In  $\mathbb{C}$  every non-constant polynomial has at least one root in  $\mathbb{C}$  (we say  $\mathbb{C}$  is *algebraically closed*):

**Theorem 0.17 (Fundamental theorem of algebra)** Let  $\alpha_0, \alpha_1, \dots, \alpha_n \in \mathbb{C}$  with  $n \geq 1$ ,  $\alpha_n \neq 0$  (i.e., nonzero leading coefficient) and consider the nonconstant polynomial  $p: \mathbb{C} \rightarrow \mathbb{C}$ ,

$$p(z) := \sum_{i=0}^n \alpha_i z^i.$$

Then, there are numbers  $\lambda_1, \dots, \lambda_n \in \mathbb{C}$  such that

$$p(z) = \alpha_n \prod_{i=1}^n (z - \lambda_i) = \alpha_n \cdot (z - \lambda_1) \cdot \dots \cdot (z - \lambda_n), \quad \forall z \in \mathbb{C}.$$

In particular, the  $\lambda_i$  are precisely the roots of  $p$ , i.e.,  $p(\lambda_i) = 0$  for  $i = 1, \dots, n$ .

**Example 0.18**

## 0.4.2 Summary

$$\mathbb{N} \subsetneq \mathbb{Z} \subsetneq \mathbb{Q} \subsetneq \mathbb{R} \subsetneq \mathbb{C}$$

$\mathbb{N}$	<b>Natural</b>	$\{(0), 1, 2, 3, \dots\}$	order relation
$\mathbb{Z}$	<b>Integer</b>	$\{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$	$(\mathbb{Z}, +)$ ordered, commutative group
$\mathbb{Q}$	<b>Rational</b>	$\left\{ \frac{p}{q} : p, q \in \mathbb{Z}, q \neq 0 \right\}$	$(\mathbb{Q}, +, \cdot)$ ordered field
$\mathbb{R}$	<b>Real</b>	$\mathbb{Q} \cup \{\text{limits of sequences in } \mathbb{Q}\}$	$(\mathbb{R}, +, \cdot)$ ordered and complete field
$\mathbb{C}$	<b>Complex</b>	$\{a + ib : a, b \in \mathbb{R}\}, i := \sqrt{-1}$	$(\mathbb{C}, +, \cdot)$ algebraically closed field

Most **theoretical** investigations deal with real numbers  $r \in \mathbb{R}$ .

**Numerical** computations can only be performed with *floating point numbers* (short: *floats*) with a relative error (typically  $10^{-16}$ ) in each operation.

Many of the following results hold for general fields, say  $(\mathbb{F}, +, \cdot)$ . However the only fields we will know about are the real numbers  $\mathbb{R}$  and the complex numbers  $\mathbb{C}$ ; thus we always think of  $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$ .

## 0.5 Sequences

Numerical methods often produce sequences which (in the best case) *converge* to a desired solution. Also besides this, the concept of a limiting process to *infinity* is the basis for many other notions in mathematics (differentiation/integration/...).

For simplicity, in the following we only consider sequences in  $\mathbb{R}$ . In order to have a notion of “distance” we will consider the metric

$$d: \mathbb{R} \times \mathbb{R} \rightarrow [0, +\infty), \quad d(x, y) := |x - y|,$$

where

$$|x| := \text{abs}(x) := \begin{cases} x, & \text{if } x \geq 0 \\ -x, & \text{else} \end{cases} \quad (\text{absolute value of } x).$$

In the following,  $\mathbb{R}$  can also be replaced by any set  $X$  which can be equipped with a so-called **metric**  $d$  (in math we call  $(X, d)$  a metric space).

### Example 0.19

We introduce the notion of sequence and some important related properties:

**Definition 0.20 (Sequence)** Let  $M$  be a set. Then a function  $x: \mathbb{N} \rightarrow M$  is called a sequence.

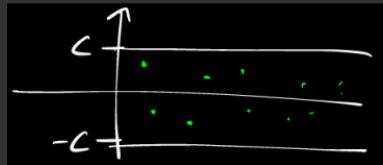
Notation:  $(x^k)_{k \in \mathbb{N}}$ ,  $\{x^k\}_{k \in \mathbb{N}}$  or  $\{x^k\}_{k=1}^\infty$ .

**Example 0.21**

**Definition 0.22** Let  $(x^k)_{k \in \mathbb{N}}$  be a sequence in  $\mathbb{R}$ . Then  $(x^k)_{k \in \mathbb{N}}$  is called

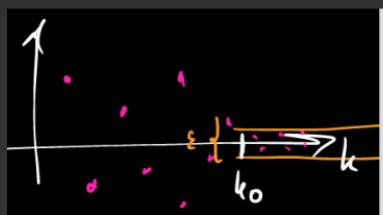
- i) **bounded**, if there exists a uniform bound  $C > 0$  such that for all  $k \in \mathbb{N}$

$$d(x^k, 0) = |x^k| \leq C.$$



- ii) **Cauchy**, if for any  $\varepsilon > 0$  there is a  $k_0 \in \mathbb{N}$  such that for all  $m, n \geq k_0$

$$d(x^m, x^n) = |x^m - x^n| < \varepsilon.$$



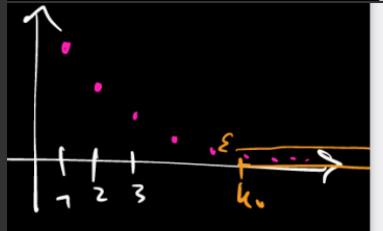
- iii) **null sequence**, if for any  $\varepsilon > 0$  there is a  $k_0 \in \mathbb{N}$  such that for all  $k \geq k_0$

$$d(x^k, 0) = |x^k| < \varepsilon.$$

We write  $\lim_{k \rightarrow \infty} x^k = 0$ .

- iv) **convergent**, if there exists a  $\bar{x} \in \mathbb{R}$  such that  $(x^k - \bar{x})_{k \in \mathbb{N}}$  is a null sequence, i.e.,  $\lim_{k \rightarrow \infty} (x^k - \bar{x}) = 0$ .

We write  $\lim_{k \rightarrow \infty} x^k = \bar{x}$ .



- v) **divergent**, if it does not converge.

### Example 0.23

**Theorem 0.24 (Uniqueness of limits)** Let  $(x^k)_{k \in \mathbb{N}}$  be a sequence in  $\mathbb{R}$ , then  $(x^k)_{k \in \mathbb{N}}$  has at most one limit.

*Proof.*

Important relation between the different types of sequences:

**Theorem 0.25** Let  $(x^k)_{k \in \mathbb{N}}$  be a sequence in  $\mathbb{R}$ , then

$$(x^k)_{k \in \mathbb{N}} \text{ convergent} \Rightarrow (x^k)_{k \in \mathbb{N}} \text{ Cauchy} \Rightarrow (x^k)_{k \in \mathbb{N}} \text{ bounded.}$$

*Proof.*

An important tool to compute limits in practice is the following:

**Theorem 0.28 (Sums and products of sequences)** Let  $(x^k)_{k \in \mathbb{N}}$  and  $(y^k)_{k \in \mathbb{N}}$  be two sequences in  $\mathbb{R}$  with  $\lim_{k \rightarrow \infty} x^k = \bar{x}$  and  $\lim_{k \rightarrow \infty} y^k = \bar{y}$ . Then

- i)  $\lim_{k \rightarrow \infty} (x^k + y^k) = \lim_{k \rightarrow \infty} x^k + \lim_{k \rightarrow \infty} y^k = \bar{x} + \bar{y};$
- ii)  $\lim_{k \rightarrow \infty} (x^k \cdot y^k) = \lim_{k \rightarrow \infty} x^k \cdot \lim_{k \rightarrow \infty} y^k = \bar{x} \cdot \bar{y}.$

In particular we find for  $a \in \mathbb{R}$  that  $\lim_{k \rightarrow \infty} (x^k + a) = \bar{x} + a$  and  $\lim_{k \rightarrow \infty} (x^k \cdot a) = \bar{x} \cdot a$ .

*Proof.* i) We define  $z^k := x^k + y^k$  and  $\bar{z} := \bar{x} + \bar{y}$ . Let  $\varepsilon > 0$ . For  $\tilde{\varepsilon} := \varepsilon/2$  there exists a  $k_0 \in \mathbb{N}$ , so that

$$|x^k - \bar{x}| < \tilde{\varepsilon} \text{ and } |y^k - \bar{y}| < \tilde{\varepsilon}$$

for all  $k \geq k_0$ . Thus by triangle equality we find

$$\begin{aligned} |z^k - \bar{z}| &= |x^k + y^k - (\bar{x} + \bar{y})| = |x^k - \bar{x} + y^k - \bar{y}| \leq |x^k - \bar{x}| + |y^k - \bar{y}| \\ &< \tilde{\varepsilon} + \tilde{\varepsilon} = \varepsilon. \end{aligned}$$

ii) We define  $z^k := x^k \cdot y^k$  and  $\bar{z} := \bar{x} \cdot \bar{y}$ . First note that by Theorem 0.25, we have that the sequence  $(x^k)_{k \in \mathbb{N}}$  is bounded, which implies that there exists a bound  $C > 0$  so that  $|x^k| < C$  for all  $k \in \mathbb{N}$ . By triangle inequality, we find

$$\begin{aligned} |z^k - \bar{z}| &= |x^k y^k - \bar{x} \bar{y}| = |x^k y^k - (x^k \bar{y} - x^k \bar{y}) + \bar{x} \bar{y}| = |x^k(y^k - \bar{y}) + (x^k - \bar{x})\bar{y}| \\ &\leq |x^k(y^k - \bar{y})| + |(\bar{x} - x^k)\bar{y}| = |x^k|(|y^k - \bar{y}|) + |(\bar{x} - x^k)||\bar{y}| \leq C(|y^k - \bar{y}|) + |(\bar{x} - x^k)||\bar{y}|. \end{aligned} \tag{1}$$

Now let  $\varepsilon > 0$ . Then for  $\tilde{\varepsilon} := \varepsilon/(C + |\bar{y}|) > 0$  there exists a  $k_0 \in \mathbb{N}$ , so that

$$|x^k - \bar{x}| < \tilde{\varepsilon} \text{ and } |y^k - \bar{y}| < \tilde{\varepsilon}$$

for all  $k \geq k_0$ . Thus, combining this with (1) gives the desired result

$$|z^k - \bar{z}| < C\tilde{\varepsilon} + \tilde{\varepsilon}|\bar{y}| = (C + |\bar{y}|)\tilde{\varepsilon} = \varepsilon.$$

### Example 0.29

- i) The sequence  $\{a + \frac{1}{k}\}_{k \in \mathbb{N}}$
- ii) The sequence  $\{a \cdot \frac{1}{k}\}_{k \in \mathbb{N}}$
- iii) The sequence  $\{\frac{1}{k^2}\}_{k \in \mathbb{N}}$



**Example 0.31** Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  with  $f(k) \geq 0$  for all  $k \in \mathbb{N}$ . Then the sequence  $\{\frac{1}{k+f(k)}\}_{k \in \mathbb{N}}$  is a null sequence.

## 0.6 Quadratic equations and completion of the square

We want to solve quadratic equations of the following kind

$$x^2 + px + q = 0, \quad (2)$$

where  $p, q \in \mathbb{R}$  are fixed. Representation (2) is sometimes referred to as **normalized form** or **reduced form**.

Recall the binomial formulas: For  $a, b \in \mathbb{R}$  we have

- i)  $(a + b)^2 = a^2 + 2ab + b^2$
- ii)  $(a - b)^2 = a^2 - 2ab + b^2$
- iii)  $(a + b)(a - b) = a^2 - b^2$

**Example 0.32** We want to find all  $x \in \mathbb{R}$ , that satisfy the quadratic equation

$$x^2 + 6x + 5 = 0.$$

In general we find:

If  $\left(\frac{p}{2}\right)^2 \geq q$ , then the quadratic equation

$$x^2 + px + q = 0$$

is solved by

$$x = -\frac{p}{2} \pm \sqrt{\left(\frac{p}{2}\right)^2 - q}.$$

*Proof.*

## Alternative representation (I)

Quadratic equations are also often introduced in the following form:

$$ax^2 + bx + c = 0 \quad (3)$$

for fixed  $a, b, c \in \mathbb{R}$ . Representation (3) is referred to as the **general form** of a quadratic equation.

Assumed  $a \neq 0$  we can deduce the reduced form (2):

**Example 0.33** We want to find all  $x \in \mathbb{R}$ , that satisfy the quadratic equation

$$2x^2 + 8x + 6.72 = 0.$$

## Alternative representation (II)

Quadratic equations are also introduced as:

$$a(x - h)^2 + k = 0 \quad (4)$$

for fixed  $a, h, k \in \mathbb{R}$ . Representation (4) is referred to as **vertex form**.

Again, we can deduce the reduced form:

## Summary

## 0.7 Inequalities

Here we make use of the fact that  $\mathbb{R}$  is an ordered field. In particular we will exploit the **monotonicity** property:

Let  $a, b \in \mathbb{R}$  with  $a \leq b$ . Then

- 1)  $\forall c \in \mathbb{R}: a + c \leq b + c$ ;
- 2)  $\forall c \geq 0: ac \leq bc$ .

From 1) and 2) it follows that

$$\forall c < 0: ac \geq bc$$

**Example 0.34** We want to find all  $x \in \mathbb{R}$ , that satisfy the inequality

$$x^2 + 6x + 5 \geq 0.$$

**Example 0.35** We want to find all  $x \in \mathbb{R}$ , that satisfy the inequality

$$2x \geq 5x - 6.$$

**Example 0.36** We want to find all  $x \in \mathbb{R}$ , that satisfy the inequality

$$\frac{x+4}{4x-12} \leq 2.$$

# Fundamentals of Linear Algebra

Recommended reading for this section:

- Lectures 1,2,3 in [4]
- Sections I.1, I.2, I.3, I.5(, I.11) in [3]
- Chapters 1,3(,4,5) in [2]

Literature:

- [1] R. Rannacher.  
*Numerik 0 - Einführung in die Numerische Mathematik.*  
Heidelberg University Publishing, 2017.
- [2] G. Strang.  
*Introduction to Linear Algebra.*  
Wellesley-Cambridge Press, 2003.
- [3] G. Strang.  
*Linear Algebra and Learning from Data.*  
Wellesley-Cambridge Press, 2019.
- [4] L.N. Trefethen and D. Bau.  
*Numerical linear algebra.*  
SIAM, Soc. for Industrial and Applied Math., Philadelphia, 1997.

# 1 Fundamentals of Linear Algebra

## 1.1 Matrices and Vectors

### Example 1.1 (*Interpolation*)

Assume we are given the following measurements

$z_i$	-1	0	1
$y_i$	0	1	0

We postulate that these measurements can be explained exactly by the (quadratic) model

$$f(z) := f_x(z) := x_1 + x_2 z^2.$$

Question: Can we find parameters  $x_1, x_2 \in \mathbb{R}$ , so that  $f(z_i) = y_i$  for all  $i = 1, \dots, 3$ ?

Throughout we will consider matrices over  $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$ . However,  $\mathbb{F}$  could be replaced by any [field](#).

### Definition 1.2 (*Matrix*)

Let  $m, n \in \mathbb{N}$ . Then a rectangular array of numbers in  $\mathbb{F}$  with  $m$  rows and  $n$  columns, written as

$$A = (a_{ij})_{ij} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix},$$

is called a  $(m \times n)$  **matrix with coefficients in  $\mathbb{F}$** .

## Operations

We can add matrices of same size and scale the entries of a matrix.

**Definition 1.4 (Summing and scaling matrices)** Let  $A, B \in \mathbb{F}^{m \times n}$  be matrices,  $m, n \in \mathbb{N}$  and  $r \in \mathbb{F}$ .

- i) **Sum of matrices:**  $+ : \mathbb{F}^{m \times n} \times \mathbb{F}^{m \times n} \rightarrow \mathbb{F}^{m \times n}$

The sum  $C := A + B$  of the two matrices  $A$  and  $B$  is defined to be the matrix  $C = (c_{ij})_{ij} \in \mathbb{F}^{m \times n}$  with entries

$$c_{ij} := a_{ij} + b_{ij} \quad \text{for } i = 1, \dots, m, j = 1, \dots, n.$$

- ii) **Multiplication with scalars:**  $\cdot : \mathbb{F} \times \mathbb{F}^{m \times n} \rightarrow \mathbb{F}^{m \times n}$

The product of the matrix  $A$  with  $r \in \mathbb{F}$  is defined to be the scaled matrix

$$r \cdot A := (r \cdot a_{ij})_{ij}.$$

In this context, elements of the field  $\mathbb{F}$  are called **scalars**.



Next we provide a notation which enables us to write linear systems of equations in a concise way.  
 We recall from Example 1.1:

$$\begin{array}{rcl} 1x_1 + 1x_2 & = & 0 \\ 1x_1 + 0x_2 & = & 1 \\ 1x_1 + 1x_2 & = & 0 \end{array} \Leftrightarrow: \underbrace{\begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}}_A \cdot \underbrace{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}_x = \underbrace{\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}}_b \Leftrightarrow Ax = b$$

**Definition 1.8 (Matrix-Vector Product)** Let  $A \in \mathbb{F}^{m \times n}$  and  $x \in \mathbb{F}^n$ . Then the matrix-vector product  $b = Ax \in \mathbb{F}^m$  is defined by

$$b_i = a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n =: \sum_{\ell=1}^n a_{i,\ell}x_\ell, \quad \forall i = 1, \dots, m.$$

There are two ways of perceiving the matrix-vector product:

(1) By rows: *Used for computations*

$$\begin{pmatrix} 1 & 2 \\ 2 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1x_1 + 2x_2 \\ 2x_1 + 0x_2 \\ 0x_1 + 1x_2 \end{pmatrix} = \begin{pmatrix} \text{inner products} \\ \text{of the rows} \\ \text{with } (x_1, x_2) \end{pmatrix}$$

→ This refers to the way of computing the matrix-vector product according to “**row · column**”.

We give this type of product of two vectors a special name:

**Definition 1.10 (Inner product)** Let  $x, y \in \mathbb{F}^n$  be two vectors. Then the (standard) inner product of  $x$  and  $y$  is defined by

$$(x, y)_2 := \bar{x} \cdot y := \sum_{i=1}^n \bar{x}_i y_i = \bar{x}_1 y_1 + \cdots + \bar{x}_n y_n,$$

where  $\bar{x}_i$  denotes the complex conjugate.

(2) By columns: Used for understanding

$$\begin{pmatrix} 1 & 2 \\ 2 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_1 \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} + x_2 \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \text{linear combination} \\ \text{of the columns} \\ a_1, a_2 \end{pmatrix}$$

**Definition 1.12 (Linear combination)** Let  $a_1, \dots, a_n \in \mathbb{F}^m$ ,  $x \in \mathbb{F}^n$ . Then

$$\sum_{i=1}^n x_i a_i = x_1 a_1 + \cdots + x_n a_n = Ax \in \mathbb{F}^m$$

is called **linear combination** of the vectors  $a_1, \dots, a_n$ . Here,  $A := [a_1, \dots, a_n] \in \mathbb{F}^{m \times n}$ .

## The matrix product

We generalize the *matrix-vector* product above to a *matrix-matrix* product by observing that:

"A matrix is just a collection of columns (or vectors)."

Idea:

We make this a rigorous definition:

**Definition 1.14 (Matrix-Matrix Product)** For matrices  $A \in \mathbb{F}^{m \times r}$  and  $B \in \mathbb{F}^{r \times n}$ , we define the **matrix-matrix product** (or simply **matrix product**)  $C := A \cdot B \in \mathbb{F}^{m \times n}$  as a column wise product, i.e.,

$$\begin{pmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \dots & c_{mn} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1r} \\ a_{21} & a_{22} & \dots & a_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mr} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{r1} & b_{r2} & \dots & b_{rn} \end{pmatrix}, \text{ i.e. } \boxed{\begin{array}{l} c_{ij} = \sum_{\ell=1}^r a_{i\ell} b_{\ell j} \\ i = 1, \dots, m \\ j = 1, \dots, n \end{array}}$$



## The (conjugate) Transpose Matrix

We finally introduce the operation of transposing matrices (and vectors):

### Definition 1.16 (*Conjugate Transpose matrix*)

For a matrix  $A := (a_{ij})_{ij} \in \mathbb{F}^{m \times n}$  the **conjugate (or Hermitian) transpose matrix**  $A^H$  of  $A$  is defined as

$$A^H := (\bar{a}_{ji})_{ij} \in \mathbb{F}^{n \times m},$$

where  $\bar{a}_{ji}$  denotes the complex conjugate of the coefficient  $a_{ji}$ .

For a real matrix  $A := (a_{ij})_{ij} \in \mathbb{R}^{m \times n}$ , so that  $\bar{a}_{ji} = a_{ji}$ , this simplifies to

$$A^\top := A^H = (a_{ji})_{ij} \in \mathbb{R}^{n \times m}$$

which we then simply call the **transpose matrix**  $A^\top$  of  $A$ .



## 1.2 Span and Image – Linear Independence and Kernel

The set of all possible linear combinations or matrix–vector products is given a special name:

**Definition 1.20 (*Span and Image*)**

- i) *The span of vectors  $a_1, \dots, a_n \in \mathbb{F}^m$  is defined by*

$$\text{span}(a_1, \dots, a_n) := \left\{ \sum_{i=1}^n x_i a_i : x_i \in \mathbb{F} \right\} \subset \mathbb{F}^m.$$

*The set  $\{a_1, \dots, a_n\}$  is called **generating system** of  $\text{span}(a_1, \dots, a_n)$ .*

- ii) *The image (or column space) of a matrix  $A := [a_1, \dots, a_n] \in \mathbb{F}^{m \times n}$  is defined by*

$$\text{Im}(A) := \{Ax : x \in \mathbb{F}^n\} = \text{span}(a_1, \dots, a_n) \subset \mathbb{F}^m.$$



Let us properly define these concepts:

**Definition 1.21 (*Linear independence and kernel*)**

- i) Vectors  $a_1, \dots, a_r \in \mathbb{F}^m$  are called **linearly independent**, if the only combination that gives the zero vector is  $0a_1 + \dots + 0a_r$ .
- ii) The **kernel** of a matrix  $A \in \mathbb{F}^{m \times n}$  is defined by

$$\ker(A) := \{x \in \mathbb{F}^n : Ax = 0\},$$

i.e., the preimage of  $\{0\}$  under  $f_A$ .

We find the following important equivalent formulation of linear independence:

**Lemma 1.22** For vectors  $a_1, \dots, a_r \in \mathbb{F}^n$  we have the equivalence:

$a_1, \dots, a_r$  linearly independent  $\Leftrightarrow$  every vector  $b \in \text{span}(a_1, \dots, a_r)$  can be uniquely linearly combined from the set  $\{a_1, \dots, a_r\}$ , i.e.,  
 $\exists_{1} x_1, \dots, x_r \in \mathbb{F}: b = x_1 a_1 + \dots + x_r a_r$ .

*Remark.* This result implies the following for solutions of linear systems: Let  $x$  solve  $Ax = b$ . If  $A$  has independent columns, then the solution  $x$  is unique! On the contrary, if the columns are dependent, we will learn that there are infinitely many solutions!





## 1.3 Subspaces of $\mathbb{F}^n$ – Basis and Dimension

**Definition 1.24 (Subspace)** A subset  $V \subset \mathbb{F}^n$  is called (linear) subspace of  $\mathbb{F}^n$  if

- i) it is nonempty, i.e.,  $V \neq \emptyset$ ,
- ii) and if it is closed under linear combinations, i.e., if

$$\lambda_1 v_1 + \lambda_2 v_2 \in V \quad \text{for all } v_1, v_2 \in V, \lambda_1, \lambda_2 \in \mathbb{F}.$$

**Question:** Is it possible to describe a linear subspace of  $\mathbb{F}^n$  by a finite number of vectors?

**Definition 1.25 (Basis)** Let  $V \subset \mathbb{F}^n$  be a subspace of  $\mathbb{F}^n$ . Then a set of vectors  $\{v_1, \dots, v_r\} \subset V$  with  $r \leq n$  is called **basis of  $V$** , if

- i)  $v_1, \dots, v_r$  are linearly independent,
- ii)  $\text{span}(v_1, \dots, v_r) = V$ .





In the exercises we will prove that for any matrix  $A \in \mathbb{F}^{m \times n}$ , the kernel  $\ker(A)$  is a subspace of  $\mathbb{F}^n$  and the image  $\text{Im}(A)$  is a subspace of  $\mathbb{F}^m$ . In the context of matrices these are important spaces and we give their dimensions a special name:

**Definition 1.27 (rank and nullity)** Let  $A \in \mathbb{F}^{m \times n}$ . Then

- $\text{rank}(A) := \dim(\text{Im}(A))$  is called the (column) **rank of  $A$** ,
- $\text{nullity}(A) := \dim(\ker(A))$  is called the **nullity of  $A$** .





**Question:** Can we find a general relation between the nullity and the rank of a matrix?

**Theorem 1.29 (Dimension Formula/Rank–Nullity Theorem)** Let  $A \in \mathbb{F}^{m \times n}$ , then

$$\text{rank}(A) + \text{nullity}(A) = n.$$

## 1.4 Inverse Matrices

In general:

Consider the matrix as a mapping

$$f_A : \mathbb{F}^n \rightarrow \mathbb{F}^n, x \mapsto Ax.$$

Then by definition the *mapping*  $f_A$  is **invertible**, if there exists a mapping  $f_A^{-1} : \mathbb{F}^n \rightarrow \mathbb{F}^n$  such that for all  $x, b \in \mathbb{F}^n$  we have

$$f_A(x) = b \Leftrightarrow x = f_A^{-1}(b).$$

Inserting the definition of  $f_A$  this reads as

$$Ax = b \Leftrightarrow x = A^{-1}b.$$

Verifying this condition for all possible  $x$  and  $b$  would be an ambitious endeavor. Luckily, this condition can be rephrased into conditions solely involving the matrix  $A$ . More precisely, by inserting one into the other we obtain

Let us make this a definition.

**Definition 1.31 (Inverse matrix)** A matrix  $A \in \mathbb{F}^{n \times n}$  is called **invertible**, if there exists a matrix  $\tilde{A} \in \mathbb{F}^{n \times n}$  with

$$A \cdot \tilde{A} = \tilde{A} \cdot A = I_n. \quad (5)$$

In case of existence we find that  $\tilde{A}$  is unique (see below) and we denote by  $A^{-1} := \tilde{A}$  the **inverse matrix** of  $A$ . The set of all invertible matrices in  $\mathbb{F}^{n \times n}$  is denoted by  $GL_n(\mathbb{F})$ , the so-called general linear group.

From the dimension formula 1.29 for  $n = m$ , we find “**injectivity = surjectivity**”

**Remark:**

A System  $Ax = b$  can be solvable even if  $A$  is not squared (and thus not invertible)!



## 1.5 The Euclidean Norm

Let us first consider the 2d and 3d case:

This idea can be generalized to:

**Definition 1.33 (Euclidean Norm)** *The Euclidean norm of a vector  $x \in \mathbb{F}^n$  is defined by*

$$\|x\|_2 := \sqrt{\sum_{i=1}^n |x_i|^2} = \sqrt{x^H x}$$

where  $|a + ib|^2 := a^2 + b^2$  denotes the absolute value of a complex number. For a real vector  $x \in \mathbb{R}^n$  this simplifies to  $\|x\|_2 := \sqrt{\sum_{i=1}^n x_i^2} = \sqrt{x^\top x}$ .

→ We will also get to know other “norms” (e.g., Manhattan norm or maximum norm).

## Relating the inner product to projections

Let us consider  $\mathbb{F} = \mathbb{R}$ . As a special case of the so-called **Cauchy Schwarz inequality** one can show that, for any two real vectors  $x, y \in \mathbb{R}^n$ ,

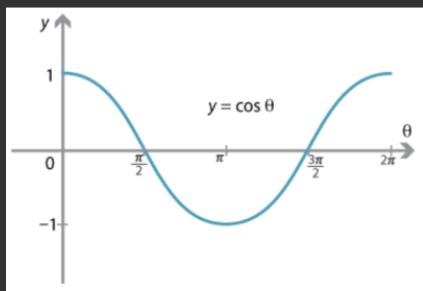
$$|x^T y| \leq \|x\|_2 \cdot \|y\|_2.$$

This is equivalent to (assumed both vectors are nonzero, otherwise trivial case)

$$-1 \leq \frac{x^T y}{\|x\|_2 \cdot \|y\|_2} = \left( \frac{x}{\|x\|_2} \right)^T \left( \frac{y}{\|y\|_2} \right) \leq 1.$$

Since  $\cos: (0, \pi) \rightarrow (-1, 1)$  is bijective, we find an uniquely defined angle  $\alpha \in (0, \pi)$ , so that

$$\cos(\alpha) = \frac{x^T y}{\|x\|_2 \cdot \|y\|_2} \quad (\in (-1, 1)).$$



We also use the notation  $\alpha := \sphericalangle(x, y)$ , since  $\alpha$  can be considered 'the angle between  $x$  and  $y$ '.

Geometric insights from the identity

$$\text{cosine} = \frac{\text{adjacent}}{\text{hypotenuse}}.$$

## 1.6 Orthogonal Vectors and Matrices

Let us again consider the relation

$$\cos(\alpha) = \frac{x^T y}{\|x\|_2 \cdot \|y\|_2}, \quad x, y \in \mathbb{R}^n.$$

Now let us assume that the angle  $\alpha = \angle(x, y)$  between the two vectors  $x, y$  is  $90^\circ$ , i.e.,  $\alpha = \pm \frac{\pi}{2}$ , meaning that they are *perpendicular*. Then we find

$$0 = \cos\left(\pm \frac{\pi}{2}\right) = \frac{x^T y}{\|x\|_2 \cdot \|y\|_2} \quad \Leftrightarrow \quad 0 = x^T y.$$

In mathematics we call this *orthogonal* and make it a general definition:

**Definition 1.34 (Orthogonal/-normal vectors)**

- i) Two vectors  $x, y \in \mathbb{F}^n$  are called **orthogonal** if  $(x, y)_2 = x^H y = 0$ .
- ii) Two vectors  $x, y \in \mathbb{F}^n$  are called **orthonormal** if they are orthogonal and have length 1 (i.e.,  $\|x\|_2 = \|y\|_2 = 1$ ).
- iii) Vectors  $x_1, \dots, x_r \in \mathbb{F}^n$  are called (mutually) **orthogonal (orthonormal)** if  $x_i, x_j$  are orthogonal (orthonormal) for all possible pairs  $i \neq j \in \{1, \dots, r\}$ .

One can show that:

$$x, y \text{ orthogonal} \quad \Rightarrow \quad x, y \text{ linearly independent.} \tag{6}$$



Now let us extend this notion to matrices:

For this purpose observe that the matrix-matrix product  $Q^H Q$  for  $Q \in \mathbb{F}^{n \times n}$  contains all possible inner products of its columns:

Let us assume that the columns of  $Q$  are mutually orthonormal, then

$$Q^H Q = I_n.$$

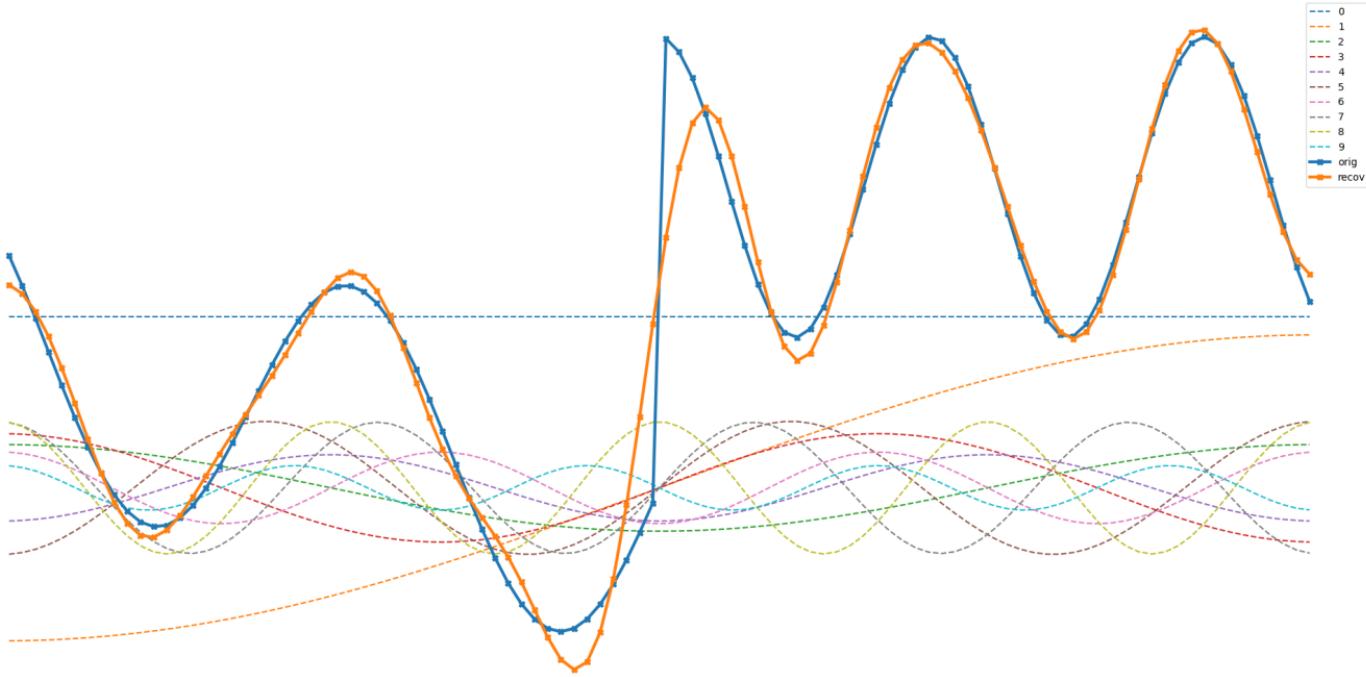
Since this is a central property, we make this a definition:

**Definition 1.36 (Orthogonal/Unitary matrix)** A matrix  $Q \in \mathbb{F}^{n \times n}$  is called **unitary**, if

$$Q^H Q = I_n.$$

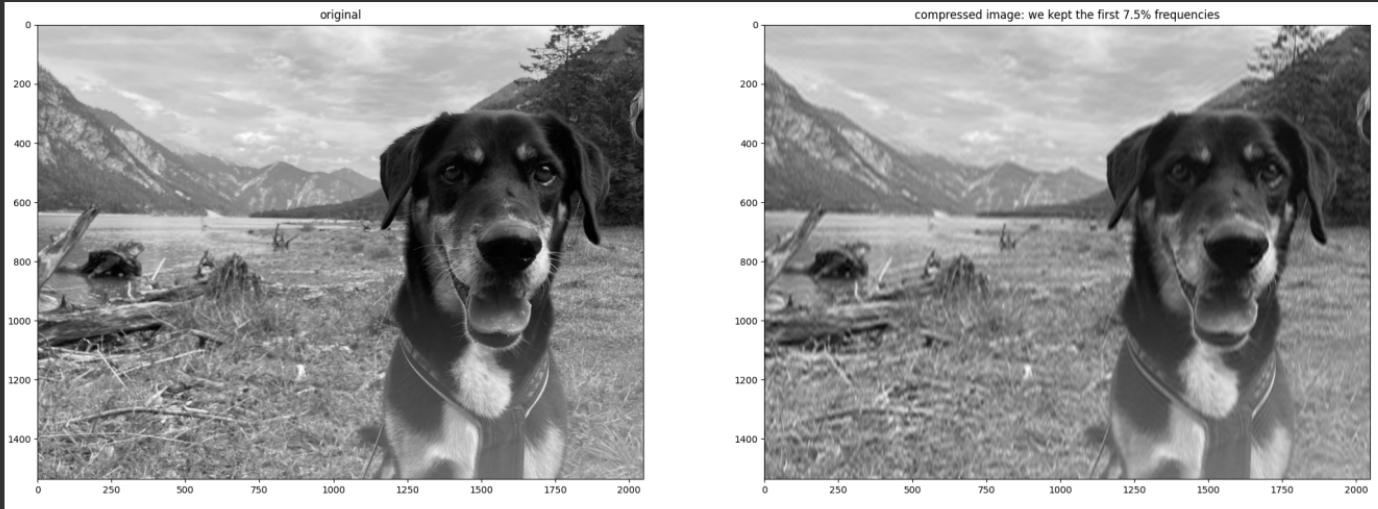
For a real matrix  $Q \in \mathbb{R}^{n \times n}$  this condition simplifies to  $Q^T Q = I_n$ , in which case we then call the matrix **orthogonal**.

## Understanding $QQ^\top(\cdot)$ as orthogonal projection



1-d DCT compression example (where high frequencies are removed):

$$y = \sum_{i=1}^n q_i^\top y \cdot q_i \approx \sum_{i=1}^m q_i^\top y \cdot q_i \quad (m < n).$$



2-d DCT compression example (where high frequencies are removed)

## 1.7 The Determinant

**Aim:** For  $n$  vectors in  $\mathbb{F}^n$  we want to have a *measure of linear independence*

- or equivalently a *volume measure* for the parallelotope spanned by these vectors
- or equivalently a *measure for the invertibility* of a matrix in  $\mathbb{F}^{n \times n}$



In general, there is the following (recursive) formula, which we use as the definition here:

**Definition 1.37 (Laplace formula)** Let  $A \in \mathbb{F}^{n \times n}$  and let  $A_{ij} \in \mathbb{F}^{(n-1) \times (n-1)}$  be the matrix resulting from erasing the  $i$ -th row and  $j$ -th column. Then the mapping  $\det: \mathbb{F}^{n \times n} \rightarrow \mathbb{F}$  defined by

$$\det(A) = \sum_{j=1}^n (-1)^{i+j} a_{ij} \det(A_{ij}), \quad \text{for a fixed but arbitrary } i \in \{1, \dots, n\},$$

is called the **determinant** (of  $A$ ), where  $\det(a) := a$  for  $a \in \mathbb{R} = \mathbb{R}^{1 \times 1}$ .

One can show: The determinant is a well-defined function, i.e., by the formula above the function  $\det(\cdot)$  assigns to each matrix  $A \in \mathbb{F}^{n \times n}$  exactly one number in  $\mathbb{F}$ .

**Laplace formula for  $n = 2$  and  $n = 3$ :**

- $n = 2$  (we fix  $i = 1$ )

- $n = 3$  : *Sarrus rule* (exercise)

One can show:

**Theorem 1.38 (Determinant properties)** *The determinant satisfies the following computational rules:*

- i)  $\forall A \in \mathbb{F}^{n \times n} : \det(A) \neq 0 \Leftrightarrow A \in GL(n, \mathbb{F})$  ( $\Leftrightarrow$  columns of  $A$  are linearly independent)
- ii)  $\forall A \in \mathbb{F}^{n \times n} : \det(A^\top) = \det(A)$
- iii) if  $A \in \mathbb{F}^{m \times m}, B \in \mathbb{F}^{m \times n}, C \in \mathbb{F}^{n \times n}$  and

$$M := \begin{pmatrix} A & B \\ 0 & C \end{pmatrix} \in \mathbb{F}^{(m+n) \times (m+n)}$$

then  $\det M = \det A \cdot \det C$

- iv)  $\forall A, A' \in \mathbb{F}^{n \times n} : \det(A \cdot A') = \det(A) \cdot \det(A')$

**Question:** Are there matrices for which the computation of the determinant is easy?

Yes, as in many other situations it turns out that orthogonal and triangular matrices are easy to treat! More precisely, we find:

**Corollary 1.39 (Triangular matrices)** Let  $U \in \mathbb{F}^{n \times n}$  be upper triangular, i.e.,

$$U = \begin{pmatrix} u_{11} & x & \cdots & x \\ 0 & u_{22} & & \vdots \\ \vdots & & \ddots & x \\ 0 & \cdots & 0 & u_{nn} \end{pmatrix}.$$

Then

$$\det(U) = u_{11} \cdot u_{22} \cdot \dots \cdot u_{nn}.$$

In particular, we find

$$U \text{ is invertible} \Leftrightarrow \det(U) \neq 0 \Leftrightarrow \forall i : u_{ii} \neq 0$$

*Proof.* Exercise: For the product formula apply Theorem 1.38 iii) inductively. The second part then easily follows from Theorem 1.38 i).

**Corollary 1.40 (Orthogonal matrices)** Let  $Q \in \mathbb{R}^{n \times n}$  be an orthogonal matrix, then  $|\det(Q)| = 1$ .

*Proof.* From Cor. 1.39 we find  $\det(I) = 1$ . Then result follows from Theorem 1.38 ii) and iv).

## 1.8 Linear Systems of Equations

Aim:

*Given  $A \in \mathbb{R}^{m \times n}$  ( $m \neq n$  possible) and  $b \in \mathbb{R}^m$ , find  $x \in \mathbb{R}^n$  such that  $Ax = b$ .*

### 1.8.1 Motivation: Curve Fitting









## 1.8.2 Existence and Uniqueness Analysis

## *Summary*

### **Aim:**

*Given  $A \in \mathbb{R}^{m \times n}$  ( $m \neq n$  possible) and  $b \in \mathbb{R}^m$ , find  $x \in \mathbb{R}^n$  such that  
 $Ax = b$ .*



## 1.9 More on Image and Kernel

Let us fix  $\mathbb{F} = \mathbb{R}$  in this section. In this subsection we derive some more results on the kernel

$$\ker(A) = \{x \in \mathbb{R}^n : Ax = 0\} \subset \mathbb{R}^n$$

and the image

$$\text{Im}(A) = \{Ax : x \in \mathbb{R}^n\} \subset \mathbb{R}^m.$$

These results prove useful in later sections; in particular when we talk about the singular value decomposition.

## The Four Fundamental Subspaces

In the context of a matrix  $A \in \mathbb{R}^{m \times n}$  there are four subspaces that stand out:

$$\ker(A) \perp \text{Im}(A^\top)$$

$$\text{Im}(A) \perp \ker(A^\top).$$

**Example 1.42** Let us consider

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 6 \end{pmatrix}, \quad A^\top = \begin{pmatrix} 1 & 3 \\ 2 & 6 \end{pmatrix}$$

We need another definition:

**Definition 1.43 (Orthogonal subspaces)** Let  $U, V \subset \mathbb{R}^n$  be two subspaces.

- i) We call  $U$  and  $V$  **orthogonal** ( $U \perp V$ ) if  $u^\top v = 0$  for all  $u \in U, v \in V$ .
- ii) We call

$$U^\perp := \{x \in \mathbb{R}^n : x^\top u = 0 \quad \forall u \in U\}$$

the **orthogonal complement** of  $V$  in  $\mathbb{R}^n$ .

*Exercise:* Show that  $(U^\perp)^\perp = U$  and  $U \perp U^\perp$ .

**Example 1.44**

We now prove the orthogonality relation between the four fundamental subspaces:

**Lemma 1.45** *Let  $A \in \mathbb{R}^{m \times n}$ . Then*

$$\text{Im}(A)^\perp = \ker(A^\top) \quad \text{and} \quad \ker(A)^\perp = \text{Im}(A^\top).$$

*In words,  $\ker(A^\top)$  is the orthogonal complement of  $\text{Im}(A)$  in  $\mathbb{R}^m$  and  $\text{Im}(A^\top)$  is the orthogonal complement of  $\ker(A)$  in  $\mathbb{R}^n$ .*

In terms of the transpose matrix we find two more characterizations of the image and kernel:

**Lemma 1.46** *Let  $A \in \mathbb{R}^{m \times n}$ . Then*

- i)  $\ker(A) = \ker(A^\top A)$  (and  $\ker(A^\top) = \ker(AA^\top)$ ),
- ii)  $\text{Im}(A) = \text{Im}(AA^\top)$  (and  $\text{Im}(A^\top) = \text{Im}(A^\top A)$ ).

### Remark

The so-called Gram matrix  $A^\top A$  plays a crucial role in many applications and also analysis, for instance

- it plays a key role to derive the singular value decomposition
- it is the system matrix in the normal equation  $A^\top Ax = A^\top x$  for solving least squares problems
- in graph theory it appears as graph Laplacian
- if  $A \approx \nabla$  (gradient), then  $A^\top \approx \text{div}$  (divergence) and  $A^\top A \approx \Delta$  (Laplacian)

A generalization of this result is given by the following lemma.

**Lemma 1.47** *Let  $A \in \mathbb{R}^{m \times n}$ . Then*

- i) *For a matrix  $B \in \mathbb{R}^{\ell \times m}$  with  $\ker(B) = \{0\}$  ("injective") we have*

$$\ker(BA) = \ker(A).$$

- ii) *For a matrix  $C \in \mathbb{R}^{n \times k}$  with  $\text{Im}(C) = \mathbb{R}^n$  ("surjective") we have*

$$\text{Im}(AC) = \text{Im}(A).$$

## Example

The typical context to apply Lemma 1.47 occurs when we have a decomposition of a matrix  $A$  and want to investigate its kernel and its image.

For example, consider the reduced QR decomposition  $A = QR$ , where  $Q \in \mathbb{R}^{m \times n}$  contains orthonormal columns and  $R \in \mathbb{R}^{n \times n}$  is upper triangular. Suppose that  $A$  has full rank, i.e.,  $\text{rank}(A) = n$ , so that  $R$  is invertible (in particular  $\text{rank}(R) = n$  and  $\ker(R) = \{0\}$ ). We thus find by Lemma 1.47 i) that

$$\ker(A) = \ker(QR) = \ker(R) = \{0\}$$

and by Lemma 1.47 ii) that

$$\text{Im}(A) = \text{Im}(QR) = \text{Im}(Q).$$

With other words, the  $n$  columns in  $Q$  are an orthonormal basis for the image  $\text{Im}(A)$  of  $A$ .

## Solving Linear Systems (Direct Methods)

## Recommended reading:

- Lectures in [4]: 6,7,8 for  $QR$ ; 20,21 for  $LU$ ; 23 for  $LL^\top$
- Section I.4 in [3]
- Sections 2.6, 2.7 in [2]

## References

- [1] R. Rannacher.  
*Numerik 0 - Einführung in die Numerische Mathematik.*  
Heidelberg University Publishing, 2017.
- [2] G. Strang.  
*Introduction to Linear Algebra.*  
Wellesley-Cambridge Press, 2003.
- [3] G. Strang.  
*Linear Algebra and Learning from Data.*  
Wellesley-Cambridge Press, 2019.
- [4] L.N. Trefethen and D. Bau.  
*Numerical linear algebra.*  
SIAM, Soc. for Industrial and Applied Math., Philadelphia, 1997.

## 2 Solving Linear Systems with Direct Methods

Aim:

*Given  $A \in \mathbb{R}^{m \times n}$  ( $m \neq n$  possible) and  $b \in \mathbb{R}^m$ , find  $x \in \mathbb{R}^n$  such that  
 $Ax = b$ .*

### 2.1 The Idea of “Factor and Solve”

A general theme in numerical mathematics is to reduce the general (possibly complicated) problem to one or more simpler problems with the help of transformations for which the property of interest is an invariant.

## Separate: Factorization and Solution!

- Since the *factorization* (*elimination/triangularization/orthogonalization*) step is typically much more expensive than the *solution* step, it makes sense to perform them separately, if the same matrix  $A$  is used for multiple right-hand sides  $b$ .
- The number of factors in a decomposition is the number of systems to be solved in the *solution* step.  
From the factors we may also gain information about the
  - the image and kernel of  $A$
  - invertibility of the matrix  $A$  in case  $m = n$
  - solvability of the system (i.e., the cardinality  $|S|$  of the solution set  $S$ )
- We will have a look at the following decompositions
  - $A = QR$  (two systems to be solved)
  - $A = P^T LU$  (three systems to be solved)
  - $A = LL^T$  (two systems to be solved)

## Never compute the inverse!

- There are only very rare occasions, where an inverse matrix  $A^{-1}$  is needed. In most cases, one needs inverse matrix times a vector, i.e.,  $A^{-1}b$ .
- For computing the inverse of an  $(n \times n)$ -matrix  $A$  we have to solve  $n$  linear systems:
- Thus, never do

```
x = inv(A) @ b
```

instead of

```
x = solve(A, b).
```

## 2.2 The Gram-Schmidt Algorithm and the $QR$ decomposition

### 2.2.1 Projectors

For a fixed vector  $x \in \mathbb{R}^n \setminus \{0\}$  we have derived the orthogonal projection onto its span by

$$\text{proj}_x(y) = \frac{x}{\|x\|_2} \cdot \frac{x^\top y}{\|x\|_2} = \frac{xx^\top}{x^\top x} \cdot y =: P_x y,$$

where  $P_x := \frac{xx^\top}{x^\top x} \in \mathbb{R}^{n \times n}$ .



For any projector  $P \in \mathbb{R}^{n \times n}$  we can show:

**Lemma 2.3 (Projector Properties)** Let  $P \in \mathbb{R}^{n \times n}$  be such  $P^2 = P$ , then

- i)  $\text{Im } P = \ker(I - P)$ ,
- ii)  $\ker(P) \cap \ker(I - P) = \{0\}$ ,
- iii)  $(I - P)$  is also a projector,
- iv)  $\forall y \in \mathbb{R}^n \exists_1 v \in \text{Im}(P), r \in \text{Im}(I - P): y = v + r$ ,  
with other words, the mapping  $\text{Im}(P) \times \text{Im}(I - P) \rightarrow \mathbb{R}^n, (v, r) \mapsto v + r$  is bijective.

Remark:

In view of Lemma 2.3 iv) we say that  $\mathbb{R}^n$  is the direct sum of the subspaces  $\text{Im}(P)$  and  $\text{Im}(I - P)$ . Each vector  $y \in \mathbb{R}^n$  uniquely splits into the additive components  $v$  and  $r$ . Due to i) and iii) of this lemma, also the same is true for the subspaces  $\text{Im}(P)$  and  $\ker(P)$ .

## Orthogonal Projection with Orthonormal Basis

Let us now consider more than just one vector. In fact, let  $q_1, \dots, q_r \in \mathbb{R}^m$  be orthonormal; thus  $r \leq m$ . Then let us define the matrices

$$\begin{aligned}\hat{Q} &:= [q_1, \dots, q_r] \in \mathbb{R}^{m \times r}, \\ P := P_{q_1, \dots, q_r} &:= \hat{Q}\hat{Q}^\top \in \mathbb{R}^{m \times m}.\end{aligned}$$

Attention: For  $r < m$ ,  $\hat{Q}$  is not an orthogonal matrix and thus  $P = \hat{Q}\hat{Q}^\top \neq I$  in general. However, for the Gramian matrix which collects all possible pairs of inner products we have  $\hat{Q}^\top \hat{Q} = I$  ( $\hat{Q}^\top$  is only a left-inverse).



**Example 2.6** Let us consider:

$$q_1 = \frac{1}{\sqrt{2}}(1, 1, 0)^\top, q_2 = \frac{1}{\sqrt{2}}(-1, 1, 0), y = (0, 1, 1)^\top \in \mathbb{R}^3$$

## Orthogonal Projection with Arbitrary Basis

Let  $a_1, \dots, a_n$  be linearly independent vectors in  $\mathbb{R}^m$  ( $m \geq n$ ) and let us put the vectors into a matrix  $A = [a_1, \dots, a_n] \in \mathbb{R}^{m \times n}$ .

How to define an orthogonal projector on the image of  $A$ ?

**Example 2.7** Let us consider:

$$a_1 = (1, 1, 0)^\top, a_2 = (0, 1, 0), y = (0, 1, 1)^\top \in \mathbb{R}^3.$$

We observe that  $\text{Im}(A) = \text{Im}(\widehat{Q})$ , so that we would expect the same projector as in the example above.

## 2.2.2 $A=QR$ from the (classical) Gram–Schmidt Algorithm

## Classical Gram–Schmidt Orthogonalization Algorithm

```
1  $r_{11} = \|a_1\|$ 
2  $q_1 = \frac{a_1}{r_{11}}$ 
3
4 for  $k = 2, \dots, n$  do
5   for  $\ell = 1, \dots, k-1$  do
6      $r_{\ell k} = a_k^\top q_\ell$ 
7   end
8    $\tilde{q}_k = a_k - \sum_{\ell=1}^{k-1} r_{\ell k} q_\ell = (I - P_{q_1, \dots, q_{k-1}}) a_k \in \text{Im}(\hat{Q}_{k-1})^\perp$ 
9    $r_{kk} = \|\tilde{q}_k\|$ 
10  if  $r_{kk} = 0$  then
11    pick arbitrary  $q_k \in \text{Im}(\hat{Q}_{k-1})^\perp = \ker(\hat{Q}_{k-1}^\top)$ 
12    // for example by solving  $\hat{Q}_{k-1}^\top q_k = 0$  and normalizing
13  end
14  else
15     $q_k = \frac{\tilde{q}_k}{r_{kk}}$ 
16  end
17 end
18 // For full QR:
19 Fill up columns of  $\hat{Q} := \hat{Q}_n$  with  $(m-n)$  orthonormal vectors of  $\ker(\hat{Q}_{k-1}^\top)$ 
20 Fill up rows of  $\hat{R} := (r_{ij})$  with  $(m-n)$  zero rows
```

*Observation: This algorithm successively orthogonalizes the columns of A!*

These ideas lead to

**Theorem 2.8 (QR decomposition)** Let  $A \in \mathbb{R}^{m \times n}$  with  $m \geq n$ , then there exists a matrix  $\widehat{Q} \in \mathbb{R}^{m \times n}$  with orthonormal columns and an upper triangular matrix  $\widehat{R} \in \mathbb{R}^{n \times n}$  such that

$$A = \widehat{Q}\widehat{R}.$$

We call this the reduced QR decomposition of  $A$ .

One can extend the columns of  $\widehat{Q}$  with orthonormal columns to obtain an orthogonal matrix  $Q \in \mathbb{R}^{m \times m}$  and the rows of  $\widehat{R}$  by zero rows to obtain a matrix  $R = \begin{pmatrix} \widehat{R} \\ 0 \end{pmatrix} \in \mathbb{R}^{m \times n}$  and thereby obtain

$$A = QR.$$

We call this the QR decomposition of  $A$ .

## **Example 2.9**





Let  $m \geq n$ .

### (1) Factorization

Gram-Schmidt orthogonalization algorithm, Householder reflections or Givens Rotations can be used.

*In Python*

```
Q, R = scipy.linalg.qr(A)
```

*or for the reduced version run*

```
Q, R = scipy.linalg.qr(A, mode="economic")
```

### (2) Solving

## 2.3 Gaussian Elimination and the *LU* Decomposition

## (1) Factorization: Row elementary operations

Apply transformations to  $A$  (and  $b$ ), which do not affect the solution  $x$  (more precisely the solution set  $S$  will not change), to bring  $A$  into a simple form and collect all transformations on the fly.

- the factorization **process** is known as *Gaussian Elimination*
- the **result** will be an invertible lower triangular matrix  $L$ , some sort of upper triangular matrix  $U$  and an orthogonal (more precisely a permutation) matrix  $P$ , such that

$$A = P^T \textcolor{orange}{LU}.$$

## (2) Solve: Forward/Backward substitution

$$Ax = b \Leftrightarrow Ux = L^{-1}Pb$$

### Example 1: Frobenius Matrices

$$A = \begin{pmatrix} 1 & 3 & 5 \\ 0 & 2 & 3 \\ 2 & 4 & 6 \end{pmatrix}, \quad b = \begin{pmatrix} 4 \\ 2 \\ 6 \end{pmatrix}$$
$$Ax = b \Leftrightarrow \begin{array}{lclcl} x_1 & +3x_2 & +5x_3 & = 4 \\ 2x_1 & +4x_2 & +6x_3 & = 6 \end{array}$$



## Frobenius Matrices

## Properties

**Lemma 2.10** 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 the matrix

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

satisfies:

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

## Example 2: Permutation Matrices



## Permutation Matrices

**Definition 2.11 (Permutation Matrix)** A matrix  $P \in \mathbb{R}^{m \times m}$  is called **permutation matrix**, if it has exactly one entry "1" in each row and column and zero otherwise.

In each step of the Gaussian elimination (with pivoting) we only swap two rows at a time: The matrix

$$P_{ik} = \begin{pmatrix} 1 & 0 & & \cdots & & 0 \\ 0 & \ddots & & & & \\ & & 1 & & & \\ & & & 0 & \cdots & 0 & 1 \\ & & & & 1 & & 0 \\ \vdots & & \vdots & & \ddots & & \vdots \\ & & 0 & & & 1 & \\ & & & 1 & 0 & \cdots & 0 \\ 0 & & & & \cdots & & 0 & 1 \end{pmatrix} \leftarrow i-th \\ \leftarrow k-th$$

swaps rows  $k$  and  $i$ , when multiplied from the left to a matrix.

→ We illuminate further properties in the exercises.



**Algorithm:** Elimination with row pivoting

## In-place Gaussian Elimination with Row Pivoting (for $m = n$ )

**INPUT:**  $A \in \mathbb{R}^{n \times n}$  (and  $b \in \mathbb{R}^n$ )

**OUTPUT:** LU decomposition  $PA = LU$  (and if  $Ax = b$  is uniquely solvable the solution  $x \in \mathbb{R}^n$ )

```
1 # FACTORIZATION
2 initialize piv = [1,2,...,n]
3 for j = 1,...,n - 1 do
4     # Find the j-th pivot
5      $k_j := \arg \max_{k \geq j} |a_{kj}|$ 
6     if  $a_{k_j j} \neq 0$  then
7         # Swap rows
8         A[kj,:]  $\leftrightarrow$  A[j,:]
9         # by hand we also transform b on the fly
10        b[kj]  $\leftrightarrow$  b[j]
11        piv[kj]  $\leftrightarrow$  piv[j]
12        # Elimination
13        for k = j + 1,...,n do
14             $\ell_{kj} := a_{kj} / a_{jj}$ 
15             $a_{kj} = \ell_{kj}$ 
16            for i = j + 1,...,n do
17                 $a_{ki} = a_{ki} - \ell_{kj} a_{ji}$ 
18            end
19            # by hand we also transform b on the fly
20             $(b_k = b_k - \ell_{kj} b_j)$ 
21        end
22    end
23 end
24 # SOLVE
25 ...
```

As in the algorithm consider  $A \in \mathbb{R}^{n \times n}$ . Then the algorithm eventually leads to

$$U := A^{(n)} = L_{(n-1)} P_{(n-1)k_{(n-1)}} \dots L_3 P_{3k_3} L_2 P_{2k_2} L_1 P_{1k_1} A.$$

By construction of the algorithm,  $A^{(n)} =: U$  has row echelon form (no rigorous proof provided in this course).

**Question:** Can we group all  $L_i$  and  $P_{ik_i}$  in such a way that  $PA = LU$ ?

**Lemma 2.13** Let  $m \in \mathbb{N}$ . 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 of the identity matrix in  $\mathbb{R}^{m \times m}$ , where  $k_i \geq i$ . Further for  $\ell_j := (0, \dots, 0, \ell_{j+1,j}, \dots, \ell_{m,j})^\top \in \mathbb{R}^m$  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

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

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

By applying this result multiple times, we obtain

## Summary: Row echelon form and LU decomposition

### Definition 2.14 (REF)

a) **Row elementary operations** are

- 1) add a nonzero multiple of one row to another
- 2) swap two rows

b) **A matrix is in row echelon form (REF)** when it satisfies the following conditions:

- 1) The first non-zero element in each row (called the leading entry) is in a column to the right of the leading entry in the previous row.
- 2) Rows with all zero elements, if any, are below rows having a non-zero element.

By applying these types of operations we find:

**Theorem 2.15 (LU Decomposition)** Every matrix  $A \in \mathbb{F}^{m \times n}$  can be transformed to REF by row elementary operations. Thus there is a matrix  $U \in \mathbb{F}^{m \times n}$  in REF, a lower triangular matrix  $L \in GL(m, \mathbb{F})$  with  $\ell_{ij} = 0, \forall j > i$ , and  $\ell_{ii} = 1, \forall i$ , and a permutation matrix  $P \in GL(m, \mathbb{F})$  with exactly one entry "1" in each row and column and zero otherwise, such that

$$P \cdot A = L \cdot U.$$

Since triangular matrices are invertible if and only if the diagonal elements are nonzero, we find for the square case  $m = n$ , that:

**Corollary 2.16 (Invertibility of A)** A matrix  $A \in \mathbb{F}^{n \times n}$  is invertible, if and only if its REF  $U \in \mathbb{F}^{n \times n}$  has a nonzero diagonal, i.e.,  $u_{ii} \neq 0, \forall i = 1, \dots, n$ .

(1) Factorization:  $\text{Lu, Piv} = \text{scipy.linalg.lu_factor}(A)$

- determine factors  $L, U$  and permutation matrix  $P$  such that  $LU = PA$
- the factors  $L, U$  are compactly stored in one matrix  $\text{Lu} \in \mathbb{R}^{n \times n}$  of the same size as  $A$  and the permutation matrix  $P$  in CSR format as integer vector  $\text{Piv} \in \mathbb{N}^n$ .

(2) Solution:  $x = \text{scipy.linalg.lu_solve}((\text{Lu}, \text{Piv}), b)$

(2.1) permute right-hand side  $\bar{b} = Pb$

(2.2) solve lower triangular system  $Lz = \bar{b}$  for  $z$  (forward substitution)

(2.3) solve REF system  $Ux = z$  for  $x$  (backward substitution)

Both, (1) and (2), are combined in the routine

$x = \text{scipy.linalg.solve}(A, b).$

### 2.3.1 Identify the number of solutions from the REF

## 2.4 The Cholesky Decomposition

For symmetric and positive definite matrices  $A \in \mathbb{R}_{\text{spd}}^{n \times n} \subset \text{GL}_n(\mathbb{R})$  we obtain the following improvement:

**Theorem 2.17** *We have the equivalence*

$$A \in \mathbb{R}_{\text{spd}}^{n \times n} \Leftrightarrow \exists_1 L \in \mathbb{R}^{n \times n} \text{ lower triangular, } l_{ii} > 0: A = LL^\top.$$

- The Decomposition  $A = LL^\top$  is called the **Cholesky decomposition of  $A$** .
- Named after the french Mathematician André-Louis Cholesky (1875-1918) who developed this decomposition during his surveying work to solve the normal equation  $A^\top Ax = A^\top b$ .
- We can derive an algorithm to compute the factor  $L$ .
- **Solving using  $A = LL^\top$**

$$Ax = b \Leftrightarrow LL^\top x = b \Leftrightarrow Lz = b, L^\top x = z \quad (\text{forward/backward Subst.})$$

*In Python:*

(1) **Factorization:** `L, lower = scipy.linalg.cho_factor(A)`

(2) **Solution:** `x = scipy.linalg.cho_solve((L, lower), b)`

## Numerical Comparison: LU vs. Cholesky

- **Computational Costs:**

The Cholesky decomposition is roughly twice as fast as Gaussian elimination (in terms of number of floating point operations). Clearly, we only need to compute one instead of two factors.

- **Stability (=“robustness against rounding errors”)** :

- Gaussian Elimination: Is only stable if a pivoting strategy is applied.
- Cholesky: Is stable even without pivoting.

(To avoid square roots, one can compute the LDL decomposition)

All in all:

*For symmetric and positive definite matrices (of moderate size),  
the Cholesky factorization is the preferred algorithm!*

## Eigenvalues: Theory and Algorithms

## Recommended reading:

- Lectures 24, 25, 27 in [4]
- Sections I.6 in [3]
- Sections 6.1, 6.2, 6.4 in [2]
- Kapitel 7 in [1]

## Literature:

[1] R. Rannacher.

*Numerik 0 - Einführung in die Numerische Mathematik.*

Heidelberg University Publishing, 2017.

[2] G. Strang.

*Introduction to Linear Algebra.*

Wellesley-Cambridge Press, 2003.

[3] G. Strang.

*Linear Algebra and Learning from Data.*

Wellesley-Cambridge Press, 2019.

[4] L.N. Trefethen and D. Bau.

*Numerical linear algebra.*

SIAM, Soc. for Industrial and Applied Math., Philadelphia, 1997.

# 3 Eigenvalues: Theory and Algorithms

## 3.1 Introduction

### 3.2 Eigenvalues and Eigendecomposition

**Definition 3.2 (Eigenvalues and -vectors)** Let  $A \in \mathbb{F}^{n \times n}$  be a matrix. A number  $\lambda \in \mathbb{C}$  is called an **eigenvalue** of  $A$ , if

$$\exists v \in \mathbb{F}^n, v \neq 0: Av = \lambda v.$$

In that case,  $v$  is called an **eigenvector** and  $(\lambda, v)$  an **eigenpair**. The set of all eigenvalues is denoted by

$$\sigma(A) := \{\lambda \in \mathbb{C}: \lambda \text{ is eigenvalue of } A\}$$

and called the **spectrum of  $A$** .

- 1) Assume we knew an eigenvalue  $\lambda$ :

- 2) Assume we had an eigenvector  $v$ :

## The determinant and eigenvalues

Let  $A \in \mathbb{F}^{n \times n}$ . Then:





### **Lemma 3.4 (Matrix and Eigenvalue Properties)**

- i) *Power of a matrix:*  $A \in \mathbb{F}^{n \times n}$ ,  $\lambda \in \sigma(A) \Rightarrow \lambda^k \in \sigma(A^k)$  for any  $k \in \mathbb{N}$
- ii) *Inverse matrix:*  $A \in GL_n(\mathbb{F})$ ,  $\lambda \in \sigma(A) \Rightarrow \lambda \neq 0$ ,  $\frac{1}{\lambda} \in \sigma(A^{-1})$
- iii) *Scaling:*  $A \in \mathbb{F}^{n \times n}$ ,  $\lambda \in \sigma(A) \Rightarrow \alpha\lambda \in \sigma(\alpha A)$  for any  $\alpha \in \mathbb{F}$
- iv)  $A \in \mathbb{F}^{n \times n}$  hermitian ( $A = A^H$ )  $\Rightarrow \sigma(A) \subset \mathbb{R}$ .
- v)  $Q \in \mathbb{F}^{n \times n}$  unitary ( $Q^H Q = I$ ),  $\lambda \in \sigma(Q) \Rightarrow |\lambda| = 1$
- vi)  $A \in \mathbb{F}^{n \times n}$  positive definite (semi-definite) ( $x^H A x > 0$  ( $\geq 0$ ))  $\Leftrightarrow \forall \lambda \in \sigma(A): \lambda > 0$  ( $\lambda \geq 0$ )
- vii) *The eigenvalues of an upper (lower) triangular matrix are sitting on the diagonal.*
- viii) *Similarity transformation:*  $A \in \mathbb{F}^{n \times n}$ ,  $T \in GL_n(\mathbb{F}) \Rightarrow \sigma(A) = \sigma(T^{-1}AT)$
- ix) *Shifts:*  $A \in \mathbb{F}^{n \times n}$ ,  $(\lambda, v)$  eigenpair of  $A \Rightarrow \forall s \in \mathbb{F}: (\lambda + s, v)$  eigenpair of  $A + sI$

**Attention:** The following rules do not hold in general:

- $\lambda \in \sigma(A), \mu \in \sigma(B) \not\Rightarrow (\lambda + \mu) \in \sigma(A + B)$
- $\lambda \in \sigma(A), \mu \in \sigma(B) \not\Rightarrow (\lambda \cdot \mu) \in \sigma(A \cdot B)$

*Proof.* Exercise.

## Diagonalizing a matrix

Let us first revisit the example from above (see Examples ?? and ??)

In the previous Example ?? the matrix  $V$  of eigenvectors turned out to be orthogonal. The next theorem states, that this is true for any real symmetric matrix.

**Theorem 3.6 (Eigendecomposition of real symmetric matrices)** For any symmetric matrix  $A \in \mathbb{R}^{n \times n}$ , there is an orthogonal matrix  $Q \in \mathbb{R}^{n \times n}$  (i.e.,  $Q^\top Q = I$ ) such that

$$Q^\top A Q = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} =: \text{diag}(\lambda_1, \dots, \lambda_n) \quad (= \text{diagonal matrix})$$

and  $\lambda_i \in \mathbb{R}, i \in \{1, \dots, n\}$ , are the eigenvalues of  $A$ . The columns of  $Q$  are the normalized eigenvectors.

*Proof.* In the exercises we will prove this statement for the special case that the matrix has  $n$  distinct eigenvalues. The general proof is rather technical and can be found in any standard textbook.

→ **Thus:** “knowing the eigenpairs = knowing the matrix”

An immediate consequence of Theorem 3.6 is this:

**Corollary 3.7** A symmetric matrix  $A \in \mathbb{R}^{n \times n}$  is invertible, if and only if all its eigenvalues are nonzero.



Geometry of the eigendecomposition:

### 3.3 Eigenvalue Algorithms: Solving the eigenvalue problem

Aim: Solving the *eigenvalue problem* defined by

Given  $A \in \mathbb{F}^{n \times n}$ , find eigenpairs  $(\lambda_i, v_i)$  so that, for all  $i = 1, \dots, n$ ,

$$v_i \neq 0 \text{ and } Av_i = \lambda_i v_i.$$

Sometimes we are only interested in a few eigenpairs  $(\lambda_i, v_i)$  (for example the one with largest eigenvalue in magnitude). In this case we call it a *partial* eigenvalue problem.

#### Overview

1. A first naive approach: Direct method  
→ only feasible for very small matrices:  $n \in \{2, 3, 4\}$
2. Partial eigenvalue problem: Simple iterative methods (here: The Power Method)  
→ determine a *single* eigenpair
3. A second advanced approach: General iterative method (here: The QR algorithm)  
→ determine *all* eigenpairs

### 3.3.1 A first naive approach: Direct method

**Recipe:**

- a) Eigenvalues:

Solving **root finding problem** for the characteristic polynomial

$$\chi_A(\lambda) := \det(A - \lambda I) = 0$$

yields the eigenvalues  $\lambda_i$ .

- b) Eigenvectors:

Solving the homogeneous **linear system**

$$(A - \lambda_i I)v_i = 0$$

for each distinct  $\lambda_i$ , gives the corresponding eigenvectors  $v_i$  (or more precisely, eigenspaces).

**Example:**  $n = 2$

Consider a general  $(2 \times 2)$ -matrix  $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ .

a) **Root finding problem:**

Above, we have derived a closed formula for the determinant of a  $(2 \times 2)$ -matrix, which applied to  $A - \lambda I$  gives

$$0 = \chi_A(\lambda) = \det(A - \lambda I) = \det \left( \begin{pmatrix} a - \lambda & b \\ c & d - \lambda \end{pmatrix} \right) = (a - \lambda)(d - \lambda) - cb = \lambda^2 - (a + d)\lambda + (ad - cb)$$
$$\rightarrow \lambda_{1/2} = \frac{a + d}{2} \pm \sqrt{\left( \frac{a + d}{2} \right)^2 - (ad - cb)}.$$

b) **Linear system:**

We then have to solve

$$\begin{pmatrix} a - \lambda_i & b \\ c & d - \lambda_i \end{pmatrix} \begin{pmatrix} v_1^i \\ v_2^i \end{pmatrix} \quad \text{for } i = 1, 2.$$
$$\rightarrow v^1, v^2$$

Note: For  $n = 3$  we can proceed similarly by applying the rule of Sarrus in step a).

## Problem:

In practice, for general, potentially very large, matrices the root finding problem is infeasible, because:

$A$  with large dimension  $n \Rightarrow \chi_A$  high polynomial degree  $\Rightarrow$  high risk of rounding errors

See for example:

[https://en.wikipedia.org/wiki/Root-finding\\_algorithms#Roots\\_of\\_polynomials](https://en.wikipedia.org/wiki/Root-finding_algorithms#Roots_of_polynomials)

**Abel–Ruffini theorem** (see related discussion in [4, Theorem 25.1]):

*There are no “closed formulas” for the roots of general polynomials with degree higher than 4.*

As a consequence:

We cannot solve the eigenvalue problem in finitely many steps.

Instead, any eigenvalue algorithm has to be iterative!

### 3.3.2 Simple Iterative Method: The Power Iteration

→ basis for PageRank algorithm from Google and the WTF algorithm from Twitter

**Theorem 3.10 (Convergence of power iteration)** Let  $A \in \mathbb{R}^{n \times n}$  be a matrix with eigenvalues  $\lambda_i$  for  $i \in \{1, \dots, n\}$  which satisfy  $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$  and whose eigenvectors form a basis of  $\mathbb{R}^n$ . Also, let the sequence of vectors  $\{w^k\}_{k=0}^\infty$  be defined by the so-called **power iteration**

$$w^{k+1} := \frac{Aw^k}{\|Aw^k\|_p}, \quad k \geq 0, p \geq 1, \quad \text{with } w^0 \text{ such that } (v^1, w^0)_2 \neq 0,$$

where  $v^1$  is the normalized (i.e.,  $\|v^1\|_p = 1$ ) eigenvector corresponding to  $\lambda_1$ . Then, for  $k \rightarrow \infty$ , we find  $w^k \rightarrow \pm v^1$  and also the so-called Rayleigh quotients

$$\mu_k := \frac{(w^k, Aw^k)_2}{(w^k, w^k)_2} \rightarrow \lambda_1.$$

*Proof.* See, e.g., [1, Satz 7.3] or [4, Theorem 27.1].

*Remark:*

- A variant of this approach is given by the so-called **inverse power method**, which can estimate any eigenpair, assumed a good initial guess for the eigenvalue is available.
- The assumption on the eigenvectors is satisfied, e.g., for real symmetric matrices (see Theorem 3.6)
- From the proof idea one finds that the convergence speed is determined by the fraction  $\left(\frac{\lambda_2}{\lambda_1}\right)^k$  (potentially very slow)

### 3.3.3 A second advanced approach: General iterative method

Recall: (Lemma 3.4)

- a) **Similar matrices** have the same eigenvalues:

$$\sigma(A) = \sigma(T^{-1}AT) \quad \text{for } T \in GL_n(\mathbb{F}).$$

- b) **Simple matrices**: Eigenvalues of an upper triangular matrix  $U$  (e.g., a diagonal matrix) are found on its diagonal, i.e.,

$$\sigma(U) = \{u_{11}, \dots, u_{nn}\}.$$

**Recipe:**

- a) Iteratively applying **similarity transformations**  $T_k \in GL_n(\mathbb{F})$  to  $A =: A_0$  thereby producing a sequence

$$A_k = T_k^{-1}A_{k-1}T_k.$$

- b) Choose  $T_k$  so that this sequence converges to a **simple matrix** (triangular or even diagonal)

$$A_\infty := \lim_{k \rightarrow \infty} A_k.$$

→ **Question:** Choice of the  $T_k$ 's?

Requirements on the transformations  $T_k$ :

1. easily constructed from  $A_{k-1}$
2. easy to invert (e.g., orthogonal matrix)
3.  $(A_k)_k$  converges to something simple

One Implementation:

- a) The **QR-Algorithmn** defines such transformations  $T_k$  through

$$A_0 = A$$

for  $k = 1, \dots, \infty$  :

$$Q_k R_k := A_{k-1}$$
$$A_k := R_k Q_k$$

b) We find:  $A_k = \overline{Q}_k^T A \overline{Q}_k \xrightarrow[k \rightarrow \infty]{} U$ , where  $U$  is **(quasi) upper triangular**; given as follows:

**Theorem 3.11 (QR Algorithm)** Consider a matrix  $A \in \mathbb{R}^{n \times n}$  with distinct eigenvalues  $\lambda_i$  for  $i = 1, \dots, n$ , i.e.,  $|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$ . Then the iterates  $A_k \in \mathbb{R}^{n \times n}$  produced by the QR algorithm converge to the diagonal matrix  $\Lambda := \text{diag}(\lambda_1, \dots, \lambda_n)$  which consists of the eigenvalues of  $A$ , i.e.,

$$\lim_{k \rightarrow \infty} A_k = \Lambda.$$

*Proof.* See, e.g., [1, Satz 7.8].

Finally: **What about the eigenvectors?**

One can further show that similar to the power iteration, we find that the columns of

$$\overline{Q}_\infty := \lim_{k \rightarrow \infty} \overline{Q}_k$$

are normalized eigenvectors of  $A$ .

### 3.3.4 In Practice: Combined Iterative Methods

Problems:

- QR decomposition for general and very large matrices too expensive
- Exact Schur complement is not reached in finitely many steps (= many QR decompositions needed)

However:

- Any matrix can be **reduced** to a Hessenberg matrix (= simple matrix) in *finitely many* steps
- QR decomposition for this type of matrix is cheap

This leads to:

#### (3) A third state-of-the-art approach: Combined iterative methods

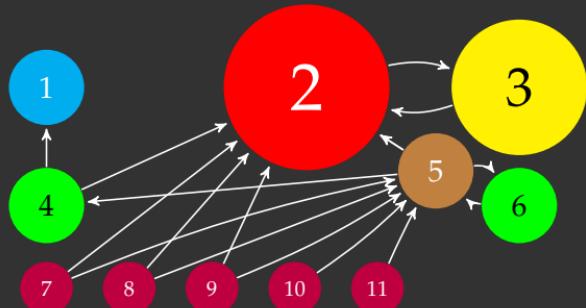
- a) **Similarity transformation via reduction** (e.g., Householder, Wilkinson, Givens) to something simple such as Hessenberg or even tridiagonal  
(→ *finite steps*)
- b) **Similarity transformation via iterative method** (e.g., QR or LR algorithm)  
(→ *potentially infinitely many steps*)  
Standard: QR Algorithm (with performance optimized modifications (shifts etc...))
- c) Determine eigenvalues from the limiting **simple matrix** (and eigenvectors from the similarity transformations).

Common combination in practice: (a) Householder reflection + (b) QR algorithm

→ Works pretty well for matrices up to 1 mio. columns  $n \approx 10^6$

→ for larger matrices one needs to develop problem-tailored structure exploiting methods

### 3.4 Example: The PageRank Algorithm from Google



**Aim:** Rank search engine results according to the “*importance*” of the web pages.

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

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

→ More details on the sheet and in the video.

## Singular Value Decomposition (SVD)

## Recommended reading:

- Lectures 4, 5 in [4]
- Sections I.8 and I.9 in [3]

## Literature:

[1] R. Rannacher.

*Numerik 0 - Einführung in die Numerische Mathematik.*

Heidelberg University Publishing, 2017.

[2] G. Strang.

*Introduction to Linear Algebra.*

Wellesley-Cambridge Press, 2003.

[3] G. Strang.

*Linear Algebra and Learning from Data.*

Wellesley-Cambridge Press, 2019.

[4] L.N. Trefethen and D. Bau.

*Numerical linear algebra.*

SIAM, Soc. for Industrial and Applied Math., Philadelphia, 1997.

# 4 Singular Values and the Singular Value Decomposition (SVD)

We will extend the concept of eigenvalues and eigenvectors to general matrices  $A \in \mathbb{R}^{m \times n}$ .

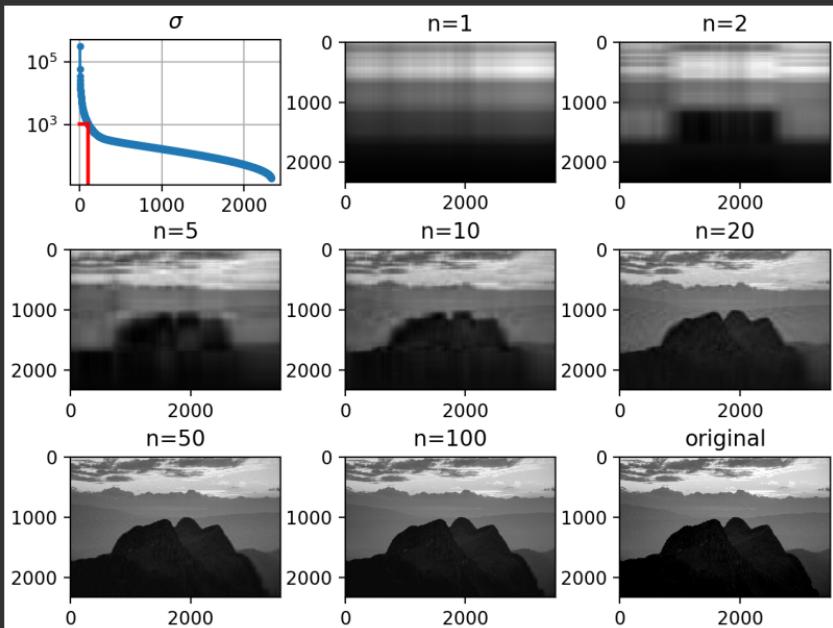
## 4.1 Motivation and Introduction

**Gilbert Strang:** “*The SVD  $A = U\Sigma V^\top$  is the **most important** theorem in data science.*”  
([3] Linear Algebra and Learning from Data, p.31)

### Importance and Applications:

- The SVD of a matrix reveals many properties about the matrix itself (representation of the image and kernel, rank, invertibility, condition,...)
- Low-Rank Approximation
  - Data compression (e.g., image data)
  - Principal Component Analysis
- Pseudoinverse (generalization of the inverse matrix) and relation to the minimum-norm least squares solution

## Image and data compression:



3500  $\times$  2333 greyscale image is interpreted as matrix

$$A \in [0, 1]^{3500 \times 2333}.$$

The singular values are shown in the figure with the title " $\sigma$ ".

The reconstructed image with the first 100 singular values only, i.e.,

$$A_{100} := U \text{diag}(\sigma_1, \dots, \sigma_{100}, 0, \dots, 0) V^\top$$

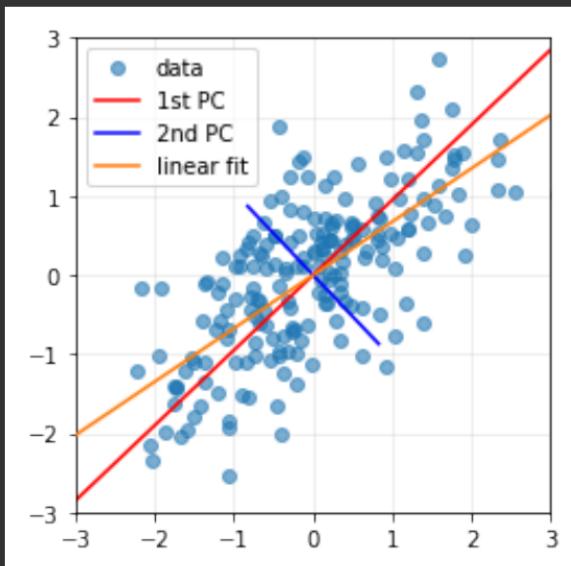
is quite close to the original image but takes only

$$\frac{3500 \cdot 100 + 100 \cdot 100 \cdot 2333}{3500 \cdot 2333} \approx 7\%$$

of the storage space.

## Principal Component Analysis

Under the correct setup we have that the SVD equals the PCA, whose aim is dimension reduction:



The data represented by the blue dots can be fully explained by the red and blue line. However the red line might already capture a substantial part of the data's variance.

## The Singular Value Decomposition (SVD)

For matrices  $A \in \mathbb{R}^{m \times n}$  of general format, the equation  $Av = \lambda v$  fails. Instead we define:

**Definition 4.1 (Singular Values and Vectors)** Let  $A \in \mathbb{R}^{m \times n}$  be a matrix. Then a positive number  $\sigma > 0$  is called **singular value**, if there exist nonzero vectors  $v \in \mathbb{R}^n \setminus \{0\}$  and  $u \in \mathbb{R}^m \setminus \{0\}$ , such that

$$Av = \sigma u \quad \text{and} \quad A^\top u = \sigma v. \quad (8)$$

The vectors  $v$  and  $u$  are called right and left **singular vectors of  $A$  to the singular value  $\sigma$** .

This will lead to the impactful theorem of the singular value decomposition:

**Theorem 4.2 (Singular value decomposition (SVD))** Let  $A \in \mathbb{R}^{m \times n}$ . Then there are orthogonal matrices  $U \in \mathbb{R}^{m \times m}$  and  $V \in \mathbb{R}^{n \times n}$  as well as a diagonal matrix  $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r, 0, \dots, 0) \in \mathbb{R}^{m \times n}$ , where  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$ ,  $r \leq \min\{m, n\}$ , are the sorted positive singular values, such that

$$A = U\Sigma V^\top,$$

which is the so-called **singular value decomposition of  $A$** .

## 4.2 Preparing Results

In order to understand and prove this central theorem we will put a few auxiliary results into position. The first one is about eigenvalues of symmetric and positive semi-definite matrices:

**Lemma 4.3 (Eigenvalues and Positivity)** *Let  $B \in \mathbb{R}^{n \times n}$  be symmetric and positive definite (semi-definite), then  $\lambda > 0$  ( $\geq 0$ ) for all eigenvalues  $\lambda \in \sigma(B)$ .*

The next result is about the shared eigenvalues of product matrices:

**Lemma 4.4 (Shared Eigenvalues of Products)** *Let  $A \in \mathbb{F}^{m \times n}$  and  $B \in \mathbb{F}^{n \times m}$ . Then the products  $AB \in \mathbb{F}^{m \times m}$  and  $BA \in \mathbb{F}^{n \times n}$  have the same nonzero eigenvalues.*

Remark:

- If  $m \neq n$ , then  $BA$  and  $AB$  have differently many eigenvalues. However the nonzero eigenvalues are the same. Thus both product matrices have at most  $\ell := \min\{m, n\}$  nonzero eigenvalues!
- In the special case that  $m = n$  and  $B$  invertible, we observe

$$B^{-1}(BA)B = (AB),$$

identifying the matrices  $AB$  and  $BA$  as being similar!

Now a special instance of the latter two results (choosing  $B = A^\top$ ) leads us to the key lemma to prove the SVD Theorem 4.2:

**Lemma 4.5** *Let  $A \in \mathbb{R}^{m \times n}$ , then the matrices  $A^\top A$  and  $AA^\top$  are symmetric, positive semi-definite and have the same positive eigenvalues.*

**Remark:**

Due to the symmetry of  $A^\top A$  and  $AA^\top$  we also know that we find orthonormal eigenvectors  $v_1, \dots, v_n$  and  $u_1, \dots, u_m$ ! The SVD will connect them!

## 4.3 From Reduced to Full SVD



## Full, Reduced and Truncated SVD

## The four fundamental subspaces revisited:

## Summary and Remarks

$$A = \left( \begin{array}{c|cc|cc|c} & & & & & \\ u_1 & \cdots & u_r & u_{r+1} & \cdots & u_m \\ \hline & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ \end{array} \right) \left( \begin{array}{ccc|cc|c} \sigma_1 & & & & & \vdots \\ & \ddots & & & & 0 & \cdots \\ & & \sigma_r & & & \vdots \\ \hline & & & & & \vdots \\ \vdots & & & & & 0 & \cdots \\ & 0 & \cdots & & & 0 & \cdots \\ \hline & & & & & \vdots & \\ \vdots & & & & & & \end{array} \right) \left( \begin{array}{ccc} - & v_1 & - \\ \vdots & \vdots & \vdots \\ - & v_r & - \\ - & v_{r+1} & - \\ \vdots & \vdots & \vdots \\ - & v_n & - \end{array} \right)$$

- we can show  $\text{Im}(A) = \text{span}(u_1, \dots, u_r)$  and  $\ker(A) = \text{span}(v_{r+1}, \dots, v_n)$ , in particular

$$\text{rank}(A) = r$$

- columns of  $V$  are orthonormal eigenvectors of  $A^\top A \in \mathbb{R}^{n \times n}$  and  $A^\top A = V(\Sigma^\top \Sigma)V^\top$
- columns of  $U$  are orthonormal eigenvectors of  $AA^\top \in \mathbb{R}^{m \times m}$  and  $AA^\top = U(\Sigma \Sigma^\top)U^\top$
- $\sigma_1^2$  to  $\sigma_r^2$  are the shared positive eigenvalues of both  $A^\top A$  and  $AA^\top$
- an SVD of the transpose  $A^\top$  is easily found by

$$A^\top = (U\Sigma V^\top)^\top = V\Sigma^\top U^\top$$

- for square matrices singular values and eigenvalues are different in general, take for example  $A = -I$
- however, for symmetric matrices  $A = Q\Lambda Q^\top$ , the singular values are the absolute values of the eigenvalues, i.e.,  $\sigma_i = \sqrt{\lambda_i^2}$  (see exercises)

## Example 4.6 (*SVD by hand*)





**Example:** rank-1 pieces

Let  $x \in \mathbb{R}^m \setminus \{0\}$  and  $y \in \mathbb{R}^n \setminus \{0\}$ , then

$$A := xy^\top = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} (y_1, \dots, y_n) = \begin{pmatrix} | & & | \\ y_1x & \cdots & y_nx \\ | & & | \end{pmatrix} \in \mathbb{R}^{m \times n}.$$

What is the SVD of  $A$ ?

## 4.4 The Geometry of the SVD

[Compare to the geometry of the eigendecomposition]

## 4.5 Matrix condition and rank

### Situation:

Let  $A = U\Sigma V^\top \in \mathbb{R}^{n \times n}$  be invertible (i.e.,  $\sigma_i \neq 0 \ \forall i$ ) and assume we want to solve  $Ax = b$ . We also assume that the data is corrupted  $\tilde{b} = b + \Delta b$  by some error  $\Delta b$ .

⇒ We obtain a perturbed solution  $\tilde{x} = x + \Delta x$  with  $\Delta x = A^{-1}\Delta b$ .

### Question:

How severe is the propagation of *data error*  $\Delta b$  to the resulting *solution error*  $\Delta x$ ?

→ Singular (eigen-) values give us this information!

**Definition 4.7 (Condition number)** Let  $A \in \mathbb{R}^{n \times n}$  be a matrix. Then we call

$$\text{cond}_2(A) := \frac{\max\{\sigma_i\}}{\min\{\sigma_i\}}$$

the **condition number** of the matrix  $A$ .

**Special Case:** Symmetric Matrices (exercise)

Let  $A \in \mathbb{R}^{n \times n}$  be a real symmetric matrix, then

$$\text{cond}_2(A) = \frac{\max\{|\lambda| : \lambda \in \sigma(A)\}}{\min\{|\lambda| : \lambda \in \sigma(A)\}}.$$

**Remark:**

If some of the singular values are actually zero or close to zero, the condition number is (almost)  $\infty$ . In this case, we cannot trust any numerical solver (for  $Ax = b$ ) in finite precision, as errors in the data  $b$  (e.g., also due to rounding errors) may severely propagate to the computed solution  $x$ .

We also call such matrices *rank deficient*.

## 4.6 The Truncated SVD and its Best Approximation Property

### Motivation:

Let the singular values be sorted  $\sigma_1 \geq \dots \geq \sigma_r > 0$ ,  $r := \text{rank}(A)$ , then the reduced SVD reads as

$$A = \sigma_1 u_1 v_1^\top + \sigma_2 u_2 v_2^\top + \dots + \sigma_i u_i v_i^\top + \dots + \sigma_{r-1} u_{r-1} v_{r-1}^\top + \sigma_r u_r v_r^\top$$

If a  $\sigma_i$  is small, then the matrix  $u_i v_i^\top$  does not contribute much to  $A$ , and similarly for  $\sigma_{i+1}, \dots, \sigma_r$ .

What about leaving them out?

This gives rise to the following definition:

**Definition 4.8 (Truncated SVD)** Let  $A = U\Sigma V^\top \in \mathbb{R}^{m \times n}$ . For  $k < r := \text{rank}(A)$  define  $\Sigma_k := \text{diag}(\sigma_1, \dots, \sigma_k) \in \mathbb{R}^{k \times k}$ ,  $U_k := [u_1, \dots, u_k] \in \mathbb{R}^{m \times k}$  and  $V_k := [v_1, \dots, v_k] \in \mathbb{R}^{n \times k}$ . Then

$$A_k := U \text{diag}(\sigma_1, \dots, \sigma_k, 0, \dots, 0) V^\top = U_k \Sigma_k V_k^\top$$

is called **truncated SVD of  $A$** .

We observe that

$$\text{rank}(A_k) = k,$$

which is why  $A_k$  is also called **rank- $k$ -approximation of  $A$** .

**Question:** Leaving out some rank-1 summands, how much do we deviate from the original matrix?

With other words: In which sense does  $A_k \in \mathbb{R}^{m \times n}$  approximate  $A \in \mathbb{R}^{m \times n}$ ?

We first need to quantify the distance between matrices, i.e., we need a *norm* for matrices in  $\mathbb{R}^{m \times n}$ !

Here we consider the so-called Frobenius norm:

If we reshape a matrix  $A \in \mathbb{R}^{m \times n}$  into a vector  $v \in \mathbb{R}^{m \cdot n}$  (e.g.,  $v_{[(j-1) \cdot m + i]} := a_{ij}$ ), then we can use our norms for vectors, e.g.,

$$\|A\|_F := \|v\|_2.$$

This is precisely:

**Definition 4.9 (Frobenius norm)** For any matrix  $A \in \mathbb{R}^{m \times n}$ , the **Frobenius norm** is defined as

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

Exercise:

- One can show that

$$\|A\|_F^2 = \text{tr}(A^\top A),$$

where  $\text{tr} :=$  “trace” denotes the sum of the diagonal entries.

- Using this fact, for  $A = U\Sigma V^\top$  with  $r = \text{rank}(A)$  we also find

$$\|A\|_F^2 = \sum_{i=1}^r \sigma_i^2.$$

Finally, the truncated SVD satisfies a best approximation property:

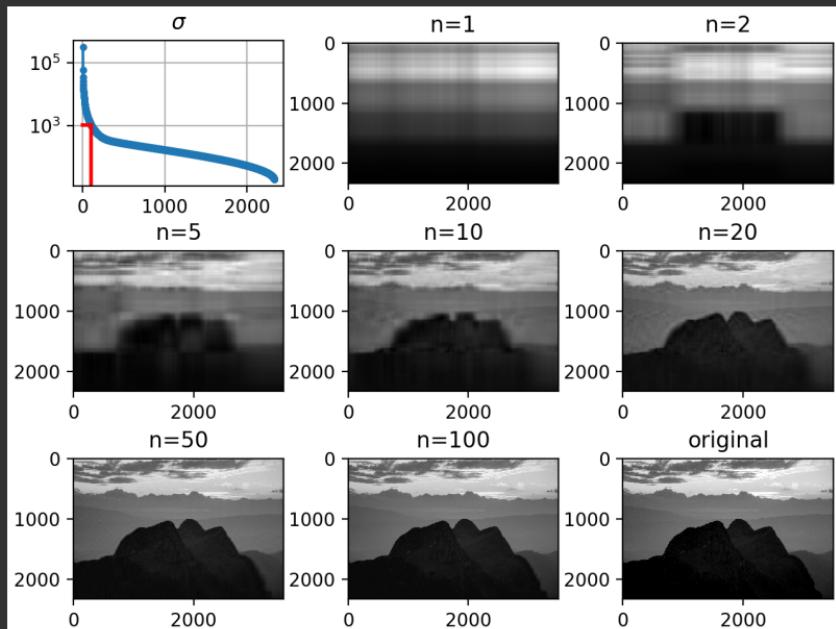
**Theorem 4.10 (Eckart-Young-Mirsky)** Let  $A \in \mathbb{R}^{m \times n}$  with SVD  $A = U\Sigma V^\top$  and let  $k \leq \text{rank}(A)$ . Then, the truncated SVD  $A_k$  is the best approximation in the Frobenius norm among all matrices with rank  $k$ , i.e.

$$\|A - A_k\|_F \leq \|A - B\|_F, \quad \forall B \in \mathbb{R}^{m \times n}, \text{rank}(B) = k.$$

In words:

*Among all matrices with rank  $k$ , the truncated SVD is closest to  $A$ .*

#### 4.6.1 Image and Data Compression



Note: The storage of  $A_k$  in general is  $k \cdot (m + 1 + n)$ .

Note: The same data compression can be performed with any matrix — and similarly with tensors.

$3500 \times 2333$  greyscale image is interpreted as matrix

$$A \in [0, 1]^{3500 \times 2333}.$$

The singular values are shown in the figure with the title “ $\sigma$ ”.

The reconstructed image with the first 100 singular values only, i.e.,

$$A_{100} := U \text{diag}(\sigma_1, \dots, \sigma_{100}, 0, \dots, 0) V^\top$$

is quite close to the original image but takes only

$$\frac{3500 \cdot 100 + 100 + 100 \cdot 2333}{3500 \cdot 2333} \approx 7\%$$

of the storage space.

#### 4.6.2 Principal Component Analysis (PCA)







#### 4.6.3 Pseudoinverses

With the help of the SVD one can define a generalized concept of an inverse matrix, called the *pseudoinverse*. This is closely related to the minimum-norm least-squares solution, so that we postpone a discussion to the section on least squares.

## 4.7 Numerical Computation of the SVD



## Solving Linear Systems with Iterative Methods

# 5 Solving Linear Systems with Iterative Methods

## 5.1 Splitting Methods

**Assumption** in this section:  $A \in \mathbb{R}^{n \times n}$  is invertible, so that  $x^* = A^{-1}b$  is the unique solution.

### 5.1.1 Motivation and Overview

Problem: *The matrix  $A$  can be very large ( $n \geq 10^5$ )!*

- If  $A$  can be stored, direct methods (such as LU, QR) are very slow or even not feasible due to large byproducts.
- Often  $A$  cannot be stored, so that direct methods are not an option.

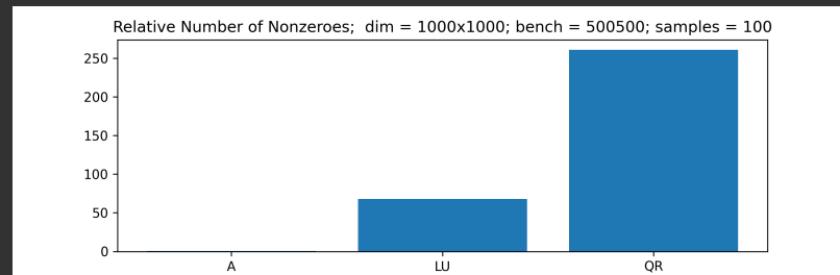
Example:

- A float with double precision (standard) needs 8 bytes of memory.
- Considering 3d measurements with just 100 measurements in each dimension.  
 $\Rightarrow 100 \cdot 100 \cdot 100 = 10^6$  measurements
- Discretization methods (e.g., finite element method) for physical models (e.g., heat diffusion) interrelate these measurements.  
 $\Rightarrow$  gives a matrix  $A \in \mathbb{R}^{n \times n}$  with  $n = 10^6$   
 $\Rightarrow n \cdot n = 10^{12}$  numbers have to be stored  
 $\Rightarrow 10^{12}$  bytes = 1 Terabyte of memory has to be allocated in RAM ↴
- A standard PC nowadays has 8–32 Gbytes of RAM (fast memory).

What if the matrix is *sparse* (= many 0 entries = redundancy)?

- We only need to store nonzero entries and their coordinates (see, e.g., CSR from previous sheets).
- **But:** Direct methods may still produce *dense* ( $\neq$  sparse) byproducts.

```
1 A = genRandomSymSparse(D)
2 # NONZEROES OF A
3 nonzeroesA += [np.count_nonzero(A)]
4 # NONZEROS OF LU
5 lu, piv = linalg.lu_factor(A)
6 nonzeroesLU += [np.count_nonzero(lu)]
7 # NONZEROS OF QR
8 Q, R = linalg.qr(A)
9 nonzeroesQR += [(np.count_nonzero(Q)+np.count_nonzero(R))]
10
11 # For sym. matrices we only need to store lower or upper triangle
12 bench = int(D*(D+1)*0.5)
13 # Average
14 nonzeroesA = np.round((np.array(nonzeroesA, dtype=float).sum()/N)/bench*100,2)
15 nonzeroesLU = np.round((np.array(nonzeroesLU, dtype=float).sum()/N)/bench*100,2)
16 nonzeroesQR = np.round((np.array(nonzeroesQR, dtype=float).sum()/N)/bench*100,2)
```



**Key idea:** We do not need the full matrix, but only some matrix-vector product  $x \mapsto S_A x$

**Approach:** Using this product we define a sequence  $\{x^0, x^1, x^2, \dots\} \subset \mathbb{R}^n$  such that

- $x^k \rightarrow x^* \in \mathbb{R}^n$  for  $k \rightarrow \infty$
- $x^*$  solves the linear equation  $Ax = b$
- $x^{k+1}$  is a better approximation to  $x^*$  than  $x^k$

Standard classes of such iterations:

- (1) **Linear iterations (splitting methods)**
- (2) **Krylov subspace methods**

**Common advantages:**

- Even a few iteration steps may yield good results in stark contrast to LU (Gaussian Elimination), which has to be performed to the bitter end.
  - The matrix  $A$  or its decomposition does not need to be stored! Only a **matrix-vector product**  $S_A x$  has to be provided.
  - In general, the overall computational complexity is lower than with *direct* methods (LU, QR,...).
- Therefore: Iterative ( $=$ *indirect*) methods are to be preferred, when the matrix is large (and sparse).

C.F. Gauß in a letter to Gerling from 1823

(<https://gdz.sub.uni-goettingen.de/id/PPN23601515X?ify>)

Fast jeden Abend mache ich eine neue Auflage des Tableaus, wo immer leicht nachzuhelfen ist. Bei der Einförmigkeit des Messungsgeschäfts gibt dies immer eine angenehme Unterhaltung; man sieht dann auch immer gleich, ob etwas zweifelhaftes eingeschlichen ist, was noch wünschenswerth bleibt, etc. Ich empfehle Ihnen diesen Modus zur Nachahmung. Schwerlich werden Sie je wieder direct eliminiren, wenigstens nicht, wenn Sie mehr als 2 Unbekannte haben. Das indirecte Verfahren lässt sich halb im Schlaf ausführen, oder man kann während desselben an andere Dinge denken.

.....



- He already mentions an iterative method which was later coined *Gauß-Seidel method*.
- In general, one may need a lot of iteration steps but the aim is to keep the iteration instruction simple and fast.

### 5.1.2 A General Framework: Linear Fixed Point Iteration

Linear iterations are of the form

$$x^{k+1} = Mx^k + Nb$$

with  $M, N \in \mathbb{R}^{n \times n}$ . The matrix  $M$  is called the **iteration matrix** and motivates the adjective “linear”.

We first derive a general convergence result and then relate  $M$  and  $N$  to the system  $Ax = b$ .

#### Convergence analysis

**Definition 5.1 (Spectral radius)** Let  $M \in \mathbb{R}^{n \times n}$ . Then the largest eigenvalue of  $M$  in magnitude is called the **spectral radius of  $M$**  and denoted by  $\rho(M)$ , more precisely

$$\rho(M) := \max\{|\lambda_1|, \dots, |\lambda_n|\}.$$

**Theorem 5.2 (Fixed point iteration)** Let  $M, N \in \mathbb{R}^{n \times n}$  and  $b \in \mathbb{R}^n$ . If  $\rho(M) < 1$ , then the sequence

$$x^{k+1} = Mx^k + Nb$$

converges for any starting point  $x^0$  and its limit  $x^* \in \mathbb{R}^n$  is a fixed point of the affine linear function  $x \mapsto Mx + Nb$ , i.e.,

$$x^* = Mx^* + Nb.$$

Proof for the special case that  $M$  is symmetric:

### 5.1.3 Splitting Methods

We now apply Theorem 5.2 to the linear system  $Ax = b$  by reformulating it as a fixed point problem. We also explain the idea of *preconditioning*.

What is a good preconditioner?

**How to construct a preconditioner?**

## General scheme:

$$x^{k+1} = (I - NA)x^k + Nb = x^k - N(Ax^k - b)$$

Preconditioner	Iteration Matrix	Iteration Instruction	Method Name
$N$	$M = I - NA$	$x^{k+1} = x^k - N(Ax^k - b)$	
$\theta I$	$M_{Rich} = I - \theta A$	$x^{k+1} = x^k - \theta(Ax^k - b)$	(relax.) <b>Richardson</b>
$\theta D^{-1}$	$M_{Jac} = I - \theta D^{-1}A$ $= (1 - \theta)I - \theta D^{-1}(L + U)$	$x^{k+1} = x^k - \theta D^{-1}(Ax^k - b)$ <i>Element-based:</i> $x_i^{k+1} = (1 - \theta)x_i^k + \frac{\theta}{a_{ii}} \left( b_i - \sum_{j \neq i} a_{ij} x_j^{k+1} \right)$	(weighted) <b>Jacobi</b>
$(L + D)^{-1}$	$M_{GS} = I - (L + D)^{-1}A$ $= (L + D)^{-1}U$	$x^{k+1} = x^k - (L + D)^{-1}(Ax^k - b)$ <i>Element-based:</i> $x_i^{k+1} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^n a_{ij} x_j^k \right)$	<b>Gauß-Seidel</b>
$\theta(\theta L + D)^{-1}$	$M_{SOR} = I - \theta(\theta L + D)^{-1}A$ $= (\theta L + D)^{-1}((1 - \theta)D - \theta U)$	$x^{k+1} = x^k - \theta(\theta L + D)^{-1}(Ax^k - b)$ <i>Element-based:</i> $x_i^{k+1} = (1 - \theta)x_i^k + \frac{\theta}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^n a_{ij} x_j^k \right)$	<b>Successive Over-Relaxation (SOR)</b>

## When does $\rho(M) < 1$ hold?

Idea: Derive some (possibly easy-to-compute) conditions which are sufficient for  $\rho(M) < 1$   
 → As usual, we often need to assume some properties for  $A$  (e.g., symmetry).

Method	Condition
<b>(relax.) Richardson</b> $M_{Rich} = I - \theta A$	<p>With relaxation (<math>\theta \neq 1</math>):</p> <p>If <math>A</math> is symmetric and positive definite (spd), then:</p> $\rho(M_{Rich}) < 1 \Leftrightarrow 0 < \theta < \frac{2}{\lambda_{max}(A)}$
<b>(weighted) Jacobi</b> $M_{Jac} = I - \theta D^{-1}A$	<p>No relaxation (<math>\theta = 1</math>):</p> <p>If <math>A</math> is strictly diagonally dominant (i.e., <math> a_{ii}  &gt; \sum_{i \neq j}  a_{ij} </math>), then <math>\rho(M_{Jac}) &lt; 1</math></p> <p>With relaxation (<math>\theta \neq 1</math>):</p> <p>If <math>A</math> is spd, then:</p> $\rho(M_{Jac}) < 1 \Leftrightarrow 0 < \theta < \frac{2}{\lambda_{max}(D^{-1}A)}$
<b>Gauß-Seidel</b> $M_{GS} = I - (L + D)^{-1}A$	<ul style="list-style-type: none"> <li>• If <math>A</math> is strictly diagonally dominant, then <math>\rho(M_{GS}) &lt; 1</math></li> <li>• If <math>A</math> is spd, then <math>\rho(M_{GS}) &lt; 1</math></li> </ul>
<b>Successive Over-Relaxation (SOR)</b> $M_{SOR} = I - \theta(\theta L + D)^{-1}A$	<p>If <math>A</math> is spd, then <math>\rho(M_{SOR}) &lt; 1</math> for <math>0 &lt; \theta &lt; 2</math></p>

**Remark:** For many matrix classes (e.g., symmetric), we find

$$\rho(M_{GS}) \leq \rho(M_{Jac}) \leq \rho(M_{Rich}) \quad (\theta = 1).$$

## Example

## Final Remarks

- Using the Richardson iteration we only need the evaluation of the matrix vector product  $x \mapsto Ax$  to solve the system  $Ax = b$ .
- All iterations of the form  $x^{k+1} = x^k - N(Ax^k - b)$  can be seen as preconditioned Richardson iterations. Next semester you will learn a strong correspondence between the Richardson method and the gradient method, as well as preconditioned Richardson method and Newton-type methods to minimize functions of the form  $f(x) := \frac{1}{2}x^\top Ax - b^\top x$ .
- By looking at the element-wise formulas for the Jacobi (with  $\theta = 1$ ) and Gauß-Seidel method (see the orange formulas in the table above) we can understand these methods as alternating methods. Consider for example the simple case  $n = 2$  and write out these formulas. Then you will see that we alternatingly compute the components  $x_1$  and  $x_2$ .
- Similar to a block LU factorization, we can consider blocks  $A_{ij}$  (= matrices) instead of numbers  $a_{ij}$  in the element-wise formulas of Jacobi and Gauß-Seidel. Thereby, one obtains the block Jacobi and block Gauß-Seidel method. These are highly related to so called additive and multiplicative Schwarz methods, respectively.
- In practice, splitting methods (or more precisely the preconditioner  $N$ ) are mainly used as preconditioners for Krylov subspace methods, which we will address in the next section.

## 5.2 Krylov Subspace Methods

### 5.2.1 Krylov Subspaces

Situation:

- $A \in GL_n(\mathbb{R})$  (invertible,  $n$  typically large, but  $A$  sparse)
- $b \in \mathbb{R}^n$

We want to solve  $Ax = b$ , but we cannot work with the matrix as a (dense) “array”. Instead we only have access to the mapping

$$x \mapsto Ax \quad (\text{matrix-vector product}).$$

Then given  $b \in \mathbb{R}^n$ , we can produce the vectors

$$b, Ab, A^2b, \dots, A^k b.$$

There is not much more to consider. Let us collect all linear combinations of these vectors and give it a name:

**Definition 5.3 (Krylov\* subspaces)** *Let  $A \in \mathbb{R}^{n \times n}$  and  $b \in \mathbb{R}^n$ , then the set*

$$K_r(A, b) := \text{span}\{b, Ab, A^2b, \dots, A^{r-1}b\}$$

*is called Krylov Subspace of order  $r \geq 1$  generated by  $A$  and  $b$ .*

In order to develop an iterative scheme based on this definition, we look deeper into the Krylov subspaces and first collect some insightful observations.

---

\* named after the Russian engineer Alexei Krylov who developed the idea in a paper published around 1931.

## Remarks

i) If  $b = 0$ , then  $A^k b = 0$  for all  $A \in \mathbb{R}^{n \times n}$  and all  $k \in \mathbb{N}$ , so that

$$K_r(A, 0) = \{0\} \quad \forall A \in \mathbb{R}^{n \times n}, r \geq 1.$$

ii) If  $b \neq 0$  and  $A = I$ , then  $A^k b = b$  for all  $k \in \mathbb{N}$ , so that

$$K_r(I, b) = \text{span}\{b\} \quad \forall b \neq 0, r \geq 1.$$

iii) If  $b$  is an eigenvector of  $A$  to the eigenvalue  $\lambda \in \sigma(A)$ , then  $A^k b = \lambda^k b$ , so that  $A^k b$  and  $b$  are linearly dependent, implying

$$K_r(A, b) = \text{span}\{b\} \quad \forall \text{ eigenvectors } b \text{ of } A.$$

iv) Insight from the power method:

Recall: Let  $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$  and  $b^T v_1 \neq 0$ , then  $\frac{A^k b}{\|A^k b\|} \rightarrow v_1$ .

Thus, for large  $k$  we have that the  $A^k b$  point into a “similar direction” – more precisely, into the direction of  $v_1$ . With other words the  $A^k b$  become more and more linearly dependent.

v) Dimension of Krylov subspaces:

- Since  $(K_r(A, b)) \subset \mathbb{R}^n$  is spanned by  $r$  vectors, we clearly have  $\dim(K_r(A, b)) \leq \min(r, n)$ .
- For  $A \in GL_n(\mathbb{R})$  one can show that

$$b, Ab, \dots, A^{r-1}b \text{ are independent} \quad \forall r \leq r_{\max},$$

where  $r_{\max}$  is the maximal dimension a Krylov subspace generated by  $A$  and  $b$  can have, i.e.,  
 $r_{\max} := \max_{s \leq n} (\dim K_s(A, b))$ .

vi) Next we state the crucial result, which forms the basis for the development of Krylov subspace methods:

**Lemma 5.4** *Let  $A \in GL_n(\mathbb{R})$  and  $b \in \mathbb{R}^n$ , then there exists an order  $r \leq n$  so that for the solution  $x^*$  of  $Ax = b$  we have*

$$x^* = A^{-1}b \in K_r(A, b).$$

*In particular, we find coefficients  $\beta_0, \dots, \beta_{r-1}$ , such that*

$$x^* = \sum_{j=0}^{r-1} \beta_j A^j b.$$

*Proof.*

## Idea of Krylov Subspace Methods

→ In this chapter we will derive the GMRES method.

## Comparison to other methods

Least squares	Krylov subspace	Splitting
$\hat{x} := \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \ Ax - b\ _2^2$ $A\hat{x} = \underset{w \in \operatorname{Im}(A)}{\operatorname{argmin}} \ w - b\ _2^2$	$x_r := \underset{x \in K_r(A, b)}{\operatorname{argmin}} \ Ax - b\ _2^2$ $Ax_r = \underset{w \in AK_r(A, b)}{\operatorname{argmin}} \ w - b\ _2^2$	$x_r := Mx_{r-1} + Nb$
$\rightsquigarrow$ Projection of $b$ onto $\operatorname{Im}(A) = A\mathbb{R}^n = \mathbb{R}^n$	$\rightsquigarrow$ Projection of $b$ onto $AK_r(A, b) = \operatorname{span}\{Ab, A^2b, \dots, A^rb\}$ $\subseteq K_{r+1}(A, b)$	
Yields exact solution, i.e., $\hat{x} = A^{-1}b$	In theory: Yields a finite sequence with $x_n = A^{-1}b$	Typically yields an infinite sequence with $x^r \xrightarrow{(k \rightarrow \infty)} A^{-1}b$

As a preparation for what follows, we will first derive an iterative method to find orthonormal bases for Krylov subspaces:

### 5.2.2 The Arnoldi Iteration

- **Situation:** Let us consider the  $r$ -th Krylov subspace

$$K_r(A, b) = \text{span}\{A^0b, A^1b, \dots, A^{r-1}b\}$$

with  $r \leq r_{\max} = \max_{s \leq n} (\dim(K_s(A, b)))$ , so that all  $A^{j-1}b$  are independent and  $\dim(K_r(A, b)) = r$ .

- **Aim:** Find an orthonormal basis  $\{q_1, \dots, q_r\}$  of  $K_r(A, b)$ .
- **Idea:** Apply the Gram–Schmidt orthogonalization process to the linearly independent vectors  $c_j := A^{j-1}b$ ,  $1 \leq j \leq r$ .
- **Recall Gram–Schmidt:** Let  $c_1, \dots, c_r \in \mathbb{R}^n$ , where  $r \leq n$ , be linearly independent vectors. Then an orthonormal basis  $\{q_1, \dots, q_r\}$  of  $\text{span}(c_1, \dots, c_r) = \text{Im}(C)$  can be found by the following iterative scheme:

$$q_1 := \frac{c_1}{\|c_1\|}$$

“subtracting projections:”  $\hat{q}_j := c_j - \sum_{\ell=1}^{j-1} q_\ell^\top c_j \cdot q_\ell$ ,  $r_{\ell j} := q_\ell^\top c_j$

“normalization:”  $q_j := \frac{\hat{q}_j}{\|\hat{q}_j\|_2}$ ,  $r_{jj} := \|\hat{q}_j\|_2$

The matrix perspective: Putting the vectors  $q_j$  and  $r_j$  (which are computed step by step) into matrices, say  $Q$  and  $R$ , then we obtain the (reduced) QR-decomposition of  $C$ , i.e., a matrix  $Q = [q_1, \dots, q_r] \in \mathbb{R}^{n \times r}$  with orthonormal columns and an upper triangular matrix  $R \in \mathbb{R}^{r \times r}$  with  $r_{ii} \neq 0$ , so that  $C = QR$ , which implies (also see Lemma 1.47)  $\text{Im}(C) = \text{Im}(QR) = \text{Im}(Q) = \text{span}(q_1, \dots, q_r)$ .

Now we consider the specific choice  $c_j := A^{j-1}b$ . By inserting these  $c_j$  into the above scheme, we obtain the so called **Arnoldi iteration**:

$$q_1 := \frac{b}{\|b\|_2} \quad (\rightarrow \text{ orthonormal basis for } K_1(A, b) = \text{span}(b), b \neq 0)$$

**for**  $j = 2, \dots, r$ :

"subtracting projections:"  $\hat{q}_j := Aq_{j-1} - \sum_{\ell=1}^{j-1} q_\ell^\top (Aq_{j-1}) \cdot q_\ell, \quad h_{\ell,j-1} := q_\ell^\top (Aq_{j-1}) \quad (*)$

"normalization:"  $q_j := \frac{\hat{q}_j}{\|\hat{q}_j\|_2}, \quad h_{j,j-1} := \|\hat{q}_j\| \quad (**)$

All in all, the Arnoldi process yields an orthonormal basis for  $K_r(A, b)$ , so that

$$K_r(A, b) = \text{span}(b, Ab, \dots, A^{r-1}b) = \text{span}(q_1, \dots, q_r)$$

or with  $Q_r := [q_1, \dots, q_r] \in \mathbb{R}^{n \times r}$  in matrix form

$$K_r(A, b) = \text{Im}(Q_r).$$

Remarks:

- Rearranging  $(*)$  and  $(**)$  easily gives

$$Aq_{j-1} = \|\hat{q}_j\|_2 \cdot q_j + \sum_{\ell=1}^{j-1} q_\ell^\top (Aq_{j-1}) \cdot q_\ell = \sum_{\ell=1}^j h_{\ell,j-1} q_\ell.$$

With other words,  $Aq_{j-1}$  is a linear combination of  $q_1, \dots, q_j$ .

- Also observe for  $j \leq r$ : The first  $q_1, \dots, q_{j-1}$  are a basis for  $K_{j-1}(A, b)$ . Thus, assumed these are given, then in order to find an orthonormal basis for  $K_j(A, b)$  we just need to compute one more vector, namely  $q_j$ .

### 5.2.3 GMRES

(with Arnoldi and  $x_0 = 0$ )

Let us recall the general idea of Krylov subspace methods: In each iteration step we compute

$$x_r := \underset{x \in K_r(A, b)}{\operatorname{argmin}} \|Ax - b\|_2^2.$$

Now we approach this minimization problem in a specific procedure resulting in the GMRES method:

We first find an orthonormal basis  $\{q_1, \dots, q_r\}$  of  $K_r(A, b)$  so that  $K_r(A, b) = \operatorname{span}(q_1, \dots, q_r) = \operatorname{Im}(Q_r)$ , where  $Q_r := [q_1, \dots, q_r] \in \mathbb{R}^{n \times r}$ . Then, since by definition  $x_r \in K_r(A, b)$ , the minimization problem can be rephrased as

$$x_r = \underset{x \in \operatorname{Im}(Q_r)}{\operatorname{argmin}} \|Ax - b\|_2^2 \stackrel{(x_r = Q_r c_r)}{=} Q_r \cdot \underbrace{\left( \underset{c \in \mathbb{R}^r}{\operatorname{argmin}} \|\textcolor{brown}{A}Q_r c - b\|_2^2 \right)}_{=: c_r, \text{ standard least squares problem}}.$$

The least squares problem can then be solved with the help of the QR-decomposition of the design matrix  $\textcolor{brown}{A}Q_r$ , say  $\tilde{Q}_r \tilde{R}_r := \textcolor{brown}{A}Q_r$ , so that the corresponding normal equation reads as

$$\tilde{R}_r c_r = \tilde{Q}_r^T b \quad (\text{when is } \tilde{R}_r \text{ invertible?})$$

Thus, GMRES boils down to the three steps. For  $1 \leq r \leq n$ , do

- Step 1: Find an orthonormal basis for  $K_r(A, b)$ .
- Step 2: Find  $\tilde{Q}_r \tilde{R}_r := \textcolor{brown}{A}Q_r$ .
- Step 3: Solve  $\tilde{R}_r c_r = \tilde{Q}_r^T b$  and set  $x_r := Q_r c_r$ .

Crucial: We can iteratively compute step 1 and step 2, i.e., use  $Q_{r-1}$ ,  $\tilde{R}_{r-1}$ ,  $\tilde{Q}_{r-1}$  to obtain  $Q_r$ ,  $\tilde{R}_r$ ,  $\tilde{Q}_r$ !

## GMRES – Step 1 “Find an orthonormal basis for $K_r(A, b)$ ”

Given  $Q_{r-1} = (q_1, \dots, q_{r-1}) \in \mathbb{R}^{n \times (r-1)}$  with  $\text{Im}(Q_{r-1}) = K_{r-1}(A, b)$  from the previous iteration step, let us compute

$$q_r := \text{Arnoldi\_step}(q_1, \dots, q_{r-1}; Aq_{r-1}).$$

(“orthogonalizing  $Aq_{r-1}$  against all  $q_1, \dots, q_{r-1}$  by subtracting projections and normalization”)

We recall the  $r$ -th Arnoldi iteration step in an algorithmic fashion:

Given  $q_1(:= \frac{b}{\|b\|}), \dots, q_{r-1}$ , then  $q_r$  is computed by:

```

 $q_r := \text{Arnoldi\_step}(q_1, \dots, q_{r-1}; v = Aq_{r-1}):$ 
    for  $\ell = 1, \dots, r-1$  do
         $h_{\ell, r-1} = q_{\ell}^T v$ 
         $v = v - h_{\ell, r-1} q_{\ell}$    (subtracting projections)
    end
     $h_{r, r-1} = \|v\|$ 
    if  $h_{r, r-1} \neq 0$  then
         $q_r = \frac{v}{h_{r, r-1}}$            (normalization)
        return  $q_r$ 
    end
return  $v$ 
```

In matrix notation this can be summed up as follows:

$$A \cdot \underbrace{\begin{pmatrix} | & & | \\ q_1 & \cdots & q_{r-1} \\ | & & | \end{pmatrix}}_{=:Q_{r-1} \in \mathbb{R}^{n \times (r-1)}} = \underbrace{\begin{pmatrix} | & & | \\ Aq_1 & \cdots & Aq_{r-1} \\ | & & | \end{pmatrix}}_{\text{Arnoldi: } Aq_j = \sum_{\ell=1}^{j+1} h_{\ell,j} q_{\ell}} = \underbrace{\begin{pmatrix} | & & | & | \\ q_1 & \cdots & q_{r-1} & q_r \\ | & & | & | \end{pmatrix}}_{=:Q_r \in \mathbb{R}^{n \times r}} \underbrace{\begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & & \\ h_{2,1} & h_{2,2} & & & \\ 0 & h_{3,2} & & & \\ \vdots & 0 & \ddots & h_{r-1,r-2} & h_{r-1,r-1} \\ 0 & 0 & 0 & h_{r,r-2} & h_{r,r-1} \end{pmatrix}}_{=:H_{r,r-1} \in \mathbb{R}^{r \times (r-1)}}$$

$$\overset{Q_{r-1}^T \cdot |}{\Leftrightarrow} Q_{r-1}^T A Q_{r-1} = Q_{r-1}^T Q_r H_{r,r-1} = \begin{pmatrix} & & 0 \\ I_{(r-1) \times (r-1)} & | & \vdots \\ & & 0 \end{pmatrix} \begin{pmatrix} & H_{r-1} \\ \hline & \dots \\ \text{row } r \text{ of } H_{r,r-1} & \end{pmatrix} = H_{r-1}$$

- We observe that  $H_{r-1} \in \mathbb{R}^{(r-1) \times (r-1)}$  has only one subdiagonal. Such matrices are called (upper) **Hessenberg matrices**.
- In particular, for  $r = n + 1$ , we find

$$Q_n^T A Q_n = H_n \in \mathbb{R}^{n \times n}.$$

With other words, with the help of the Arnoldi iteration we can find in finitely many steps (at most  $n$  steps) an upper Hessenberg  $H_n$ , which is orthogonally similar to  $A$  and thus has the same eigenvalues. That is why one can tailor the QR-algorithm (which was an algorithm to compute eigenvalues) to matrices of Hessenberg structure. The Arnoldi iteration is therefore also considered an eigenvalue algorithm. Furthermore, if  $A$  is symmetric, so is  $H_n$ , which then becomes a tridiagonal matrix.

## GMRES – Step 2 “Compute QR-decomposition $\tilde{Q}_r \tilde{R}_r := A Q_r$ ”

Again, we want to rely on the computations from the previous steps, i.e.,  $\tilde{Q}_{r-1} = [\tilde{q}_1, \dots, \tilde{q}_{r-1}] \in \mathbb{R}^{n \times (r-1)}$  and  $\tilde{R}_{r-1} = [\tilde{r}_1, \dots, \tilde{r}_{r-1}] \in \mathbb{R}^{(r-1) \times (r-1)}$  with  $\tilde{Q}_{r-1} \tilde{R}_{r-1} = A Q_{r-1}$ . Therefore, let us first observe that

$$\tilde{Q}_r \tilde{R}_r \stackrel{!}{=} A Q_r = A \cdot [Q_{r-1} | q_r] = [A Q_{r-1} | A q_r] = [\tilde{Q}_{r-1} \tilde{R}_{r-1} | A q_r] \in \mathbb{R}^{n \times r}.$$

Let us use one Arnoldi step to orthogonalize  $A q_r$  against  $\tilde{q}_1, \dots, \tilde{q}_{r-1}$  to obtain

$$\tilde{q}_r, \tilde{r}_r := \text{Arnoldi\_step}(\tilde{q}_1, \dots, \tilde{q}_{r-1}; A q_r),$$

where the vector  $\tilde{q}_r$  and the coefficients of  $\tilde{r}_r = (\tilde{r}_{1,r}, \dots, \tilde{r}_{r,r})^\top \in \mathbb{R}^r$  are computed via

$$\hat{q}_r := A q_r - \sum_{j=1}^{r-1} \tilde{r}_{j,r} \cdot \tilde{q}_j, \quad \tilde{r}_{j,r} := \tilde{q}_j^\top (A q_r), \quad \tilde{q}_r := \frac{\hat{q}_r}{\|\hat{q}_r\|_2}, \quad \tilde{r}_{rr} := \|\hat{q}_r\|_2,$$

from which we can conclude

$$A q_r = \tilde{r}_{rr} \tilde{q}_r + \sum_{j=1}^{r-1} \tilde{r}_{j,r} \tilde{q}_j = \sum_{j=1}^r \tilde{r}_{j,r} \tilde{q}_j. \quad (*)$$

Then let us define potential candidates for the sought-after QR-decomposition as follows:

$$\tilde{Q}_r := [\tilde{Q}_{r-1} | \tilde{q}_r], \quad \tilde{R}_r := \begin{pmatrix} \tilde{R}_{r-1} & | \\ \cdots & \tilde{r}_r \\ 0 \cdots 0 & | \end{pmatrix} \in \mathbb{R}^{r \times r}.$$

By  $(*)$  we can write  $A q_r = \tilde{Q}_r \tilde{r}_r$  and indeed find, as desired,

$$A Q_r = (A Q_{r-1} | A q_r) = (\tilde{Q}_{r-1} \tilde{R}_{r-1} | \tilde{Q}_r \tilde{r}_r) = \tilde{Q}_r \tilde{R}_r.$$

## GMRES – Step 3 “Solve a triangular system”

The last step is easily performed by invoking a routine to solve the upper triangular system

$$\tilde{R}_r c_r = \tilde{Q}_r^T b$$

via backward substitution. Then we find our  $r$ -th iterate by setting

$$x_r := Q_r c_r.$$

## 5.2.4 Summary and final Remarks: GMRES with Arnoldi

- i) We only need one matrix-vector product  $v \mapsto Av$  in each step! This product can be delivered to the solver GMRES as some sort of black box function.
- ii) **An initial guess**  $x_0 \neq 0$ :

Consider

$$x_r := x_0 + p_r, \quad \text{for some } x_0 \neq 0,$$

then

$$Ax_r = b \Leftrightarrow A(x_0 + p_r) = b \Leftrightarrow Ap_r = b - Ax_0 = \hat{b}.$$

Then we could solve the auxiliary system to obtain  $p := \text{GMRES}(A, \hat{b})$  and set  $x := x_0 + p$ . The above algorithm can therefore easily be extended to allow for an initial guess  $x_0$  other than the zero vector. Also note that the Krylov subspaces are then generated based on the initial residual  $\hat{b} = b - Ax_0$ .

- iii) **Preconditioning:**

Let us consider  $N \in \text{GL}_n(\mathbb{R})$ ,  $N \approx A^{-1}$ , where  $v \mapsto Nv$  is also easy to compute. Then

$$Ax = b \Leftrightarrow \underbrace{NA}_{=: \tilde{A}} x = \underbrace{Nb}_{=: \tilde{b}} \Leftrightarrow \tilde{A}x = \tilde{b}.$$

Then we could solve the equivalent system to obtain  $x := \text{GMRES}(\tilde{A}, \tilde{b})$ .

What is actually a good precondition? Quite a bit of research has been put into this question over the last decades. What one can say: Krylov subspace methods converge quickly if the eigenvalues appear in clusters away from zero.

iv) **Restarted GMRES:**

- We need to store  $Q_r := (q_1, \dots, q_r)$ ,  $\tilde{Q}_r := (\tilde{q}_1, \dots, \tilde{q}_r)$  and  $\tilde{R}_r := (\tilde{r}_1, \dots, \tilde{r}_r)$ . Thus, if the above algorithm runs many iterations, then  $Q_r, \tilde{Q}_r$  may become large (usually dense) matrices ( $n \gg 1$ ), then the advantage of the sparsity of  $A$  is lost.
- Idea: Perform only a fixed number of iterations, say  $m = 20 - 40$  and then restart GMRES with initial guess  $x_m$ . The resulting algorithm is sometimes coined GMRES( $m$ ). You will later learn about a similar idea in the context of the L-BFGS method.
- Remark: If  $A$  is symmetric and positive definite, then one can show that we do not need the whole history  $q_1, \dots, q_r$ . This is the power of the CG-method which you will investigate next semester.

v) **Another perspective and variant:** Let us consider

$$\begin{aligned} \min_{x_r \in K_r(A, b)} \|Ax_r - b\|_2^2 &= \min_{c_r \in \mathbb{R}^r} \|AQ_r c_r - b\|_2^2 = \min_{c_r \in \mathbb{R}^r} \|Q_{r+1} H_{r+1,r} c_r - b\|_2^2 \\ &\stackrel{(\#)}{=} \min_{c_r \in \mathbb{R}^r} \underbrace{\|H_{r+1,r} c_r - \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \end{pmatrix}\|_2^2}_{\text{we can evaluate this residual without computing } x_r!} \quad (*) \end{aligned}$$

$$(\#): q_1 := \frac{b}{\|b\|} \Rightarrow b = q_1 \|b\| \Rightarrow Q_{r+1}^T b = \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \end{pmatrix} \Rightarrow b = Q_{r+1} \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Let  $\widehat{Q}_{r+1}\widehat{R}_r := H_{r+1,r}$  be a QR-decomposition, then the normal equation associated to (\*) could be solved by:

$$\widehat{R}_r c_r = \widehat{Q}_{r+1}^T \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

This results in a different variant of the GMRES and the following adaptions would apply:

- In step 2: We could derive such a QR-decomposition  $\widehat{Q}_{r+1}\widehat{R}_r := H_{r+1,r}$  from our arrays by setting

$$\widehat{Q}_{r+1} := Q_{r+1}^T \widetilde{Q}_r, \quad \widehat{R}_r := \widetilde{R}_r$$

because then

$$\widetilde{Q}_r \widetilde{R}_r \stackrel{!}{=} A Q_r = Q_{r+1} H_{r+1,r} \Leftrightarrow Q_{r+1}^T \widetilde{Q}_r \widetilde{R}_r = H_{r+1,r} \Leftrightarrow \widehat{Q}_{r+1} \widehat{R}_r = H_{r+1,r} \quad (*).$$

- In step 3: Our normal equation could then be written as

$$\widetilde{R}_r c_r = \widetilde{Q}_r^T b \Leftrightarrow \widetilde{R}_r c_r = (Q_{r+1}^T \widetilde{Q}_r)^T \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \end{pmatrix} \Leftrightarrow \widehat{R}_r c_r = \widehat{Q}_{r+1} \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

All in all we obtain the following algorithm:

**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( $A, b, x_0 = 0, \text{tol} = 1e-6, \text{maxiter=None}, N = I$ ):

$b = b - Ax_0$  //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**  $r = 2, \dots, \min(n, \text{maxiter})$  **do**

//STEP 1: use Arnoldi to find column  $q_r$  by orthogonalizing  $v$  against  $q_1, \dots, q_{r-1}$

$q_r, h_{r-1} := \text{Arnoldi\_step}(Q_{r-1}; v)$  //we don't need  $h_{r-1}$

$Q_r := [Q_{r-1}, q_r]$

$v := Aq_r$

//STEP 2: use Arnoldi to find columns  $\tilde{q}_r, \tilde{r}_r$  by orthogonalizing  $v$  against  $\tilde{q}_1, \dots, \tilde{q}_{r-1}$

$\tilde{q}_r, \tilde{r}_r := \text{Arnoldi\_step}(\tilde{Q}_{r-1}; v)$

$\tilde{Q}_r := [\tilde{Q}_{r-1}, \tilde{q}_r], \tilde{R}_r := [\tilde{R}_{r-1}, \tilde{r}_r]$

//STEP 3: solve auxiliary least squares problems to obtain coordinates

$c_r := \text{solve\_triangular}(\tilde{R}_r, \tilde{Q}_r^\top b)$

$x_r := Q_r c_r$

//Attention: Evaluate the original residual here:

**if**  $\|N^{-1}(Ax_r - b)\|_2 < \text{tol}$  **then**

| break

**end**

**end**

return  $x_r + x_0$

## Least Squares Problems

## Recommended reading:

- Lecture 11 in [4]
- Section II.2 in [3]
- This handout by Homer F. Walker:  
[https://users.wpi.edu/~walker/MA3257/HANDOUTS/least-squares\\_handout.pdf](https://users.wpi.edu/~walker/MA3257/HANDOUTS/least-squares_handout.pdf)

- [1] R. Rannacher.  
*Numerik 0 - Einführung in die Numerische Mathematik.*  
Heidelberg University Publishing, 2017.
- [2] G. Strang.  
*Introduction to Linear Algebra.*  
Wellesley-Cambridge Press, 2003.
- [3] G. Strang.  
*Linear Algebra and Learning from Data.*  
Wellesley-Cambridge Press, 2019.
- [4] L.N. Trefethen and D. Bau.  
*Numerical linear algebra.*  
SIAM, Soc. for Industrial and Applied Math., Philadelphia, 1997.

# 6 Least Squares Problems

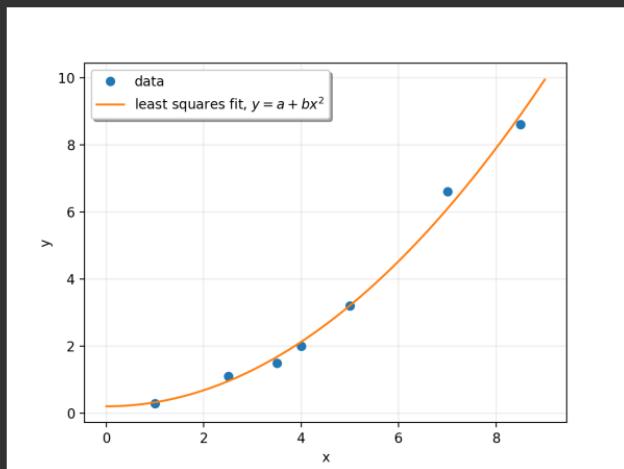
## 6.1 Overview

**Situation:** We allow for  $b \notin \text{Im}(A)$

⇒ The system  $Ax = b$  is not solvable, i.e., there is **no**  $x^* \in \mathbb{R}^n$  so that  $Ax^* = b$

**Example:** Curve fitting

The situation above typically occurs when trying to explain a set of data by just a few parameters leading to over-determined systems: more equations than unknowns ( $m \gg n$ ).



Corresponding system:

$$\begin{pmatrix} 1 & z_1^2 \\ \vdots & \vdots \\ 1 & z_n^2 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

**Approach:** Minimize the error (/residual/defect)  $\|Ax - b\|$

We obtain existence by reformulating the problem:

**Definition 6.1 (Least Squares Solution)** Let  $A \in \mathbb{R}^{m \times n}$  and  $b \in \mathbb{R}^n$ . Then  $\hat{x} \in \mathbb{R}^n$  is called a **least squares solution of  $Ax = b$** , if  $\hat{x}$  is a minimizer of the problem

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

i.e.,  $\|A\hat{x} - b\|_2^2 \leq \|Ax - b\|_2^2$  for all  $x \in \mathbb{R}^n$ .

## 6.2 The Normal Equation

The minimization problem is equivalent to a linear system:

**Theorem 6.2 (Normal Equation)** Let  $A \in \mathbb{R}^{m \times n}$  and  $b \in \mathbb{R}^m$ . Then  $\hat{x} \in \mathbb{R}^n$  is a least squares solution of  $Ax = b$  if and only if  $\hat{x}$  solves the **normal equation**

$$A^T A \hat{x} = A^T b.$$

**Proof sketch:**

(1) *Optimization perspective:*

(2) *Geometric Perspective:*

One can show that the orthogonal projection yields shortest distance:

**Lemma 6.3 (Orthogonal Projection)** *Let  $V \subset \mathbb{R}^m$  be a linear subspace and  $b \in \mathbb{R}^m$ . Then*

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

Now let us apply this lemma to our setting  $V = \text{Im}(A)$  and exploit the orthogonality of the fundamental subspaces ( $\text{Im}(A)^\perp = \ker(A^\top)$ ). We obtain

## Analysis of the Normal Equation

- (1) Properties of the system matrix  $A^T A$  (*Gramian matrix*)
  
  
  
  
  
  
  
  
- (2) For any  $A, b$  there exists a least squares solution (existence enforced ✓)

**(3)** Consistent reformulation:

If  $b \in \text{Im}(A)$ , i.e., if the original system  $Ax = b$  is solvable, then

$$\{x \in \mathbb{R}^n : Ax = b\} =: S = \widehat{S} := \{x \in \mathbb{R}^n : A^T A x = A^T b\}.$$

## 6.3 Solving the Normal Equation

**Assumptions:**

- $A$  has independent columns (existence and uniqueness ✓)
- $n$  is of moderate size (direct methods applicable ✓)

Thus there is a unique least squares solution given by (also revisit the section on projections)

$$\hat{x} = (A^T A)^{-1} A^T b.$$

**Example:** Polynomial regression

- Here we typically have many measurements  $(z_1, y_1), \dots, (z_m, y_m) \in \mathbb{R}^2$  (i.e.,  $m$  large).
- Polynomial model is given by  $f_c(z) := \sum_{j=0}^{n-1} c_j z^j$  (for  $n$  rather small because we want to smoothen the data).
- The corresponding design matrix is then given by  $A = (z_i^{j-1})_{ij}$  (revisit section on curve fitting).
- One can show:  
*If all the  $z_i$  are distinct, then the columns of  $A$  are independent (see Vandermonde matrix)!*

## Approaches:

(1) Using Cholesky decomposition  $A^T A = LL^T$

→ Problem:  $A^T A$  often *ill-conditioned* and numerical elimination may fail due to rounding errors!

Can we work without  $A^T A$ ? Yes!

(2) Using reduced QR Decomposition  $A = \widehat{Q}\widehat{R}$

### (3) Using the Pseudoinverse $A^+$ (see below)

This is typically not done in practice since the computation of the singular value decomposition (which has to be iterative in higher dimensions since we need to solve eigenvalue problems) is more expensive than a direct method. However it offers interesting theoretical insights as we will see below.

### (4) Randomized algorithms:

If  $A^T A$  is large ( $n$  large), then in particular  $A^T A$  cannot (and should not) be computed.

In such cases one can use *randomized* algorithms which only work with subsamples of the columns of  $A$  (not addressed in this course; see for example [3, II.4])

## 6.4 Regularization and Minimum Norm Least Squares Solution (enforce uniqueness)

### 6.4.1 Motivation and Overview

**Situation:** Columns of  $A$  are possibly linearly dependent ( $\ker(A) \supsetneq \{0\}$ )

$\Rightarrow A^T A$  not invertible

$\Rightarrow$  there are infinitely many solutions of  $A^T A x = A^T x$  (or if  $b \in \text{Im}(A)$ , of  $Ax = b$ )

$\rightarrow$  We say the minimization problem is *ill-posed* ( $\neq$  well-posed = existence+uniqueness)

**Question:** Which solution to pick?

We briefly discuss two approaches: **Tikhonov Regularization** and **Minimum norm solution**

## 6.4.2 Tikhonov Regularization

Tikhonov Regularization of the least squares problem (*ridge regression*):

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 + \frac{\delta}{2} \|x\|_2^2, \quad \text{for } \delta > 0 \text{ small.} \quad (14)$$

**Characterization of the “regularized” solution**

$$x_\delta := \arg \min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 + \frac{\delta}{2} \|x\|_2^2$$

**Theorem 6.4 (“Regularized” Normal Equation)** Let  $A \in \mathbb{R}^{m \times n}$  and  $b \in \mathbb{R}^m$ . Then  $x_\delta \in \mathbb{R}^n$  solves the regularized problem (14) if and only if  $x_\delta$  solves the “regularized” normal equation

$$(A^T A + \delta I)x = A^T b. \quad (15)$$

*Proof:*

## Analysis of the “regularized” normal equation

### 6.4.3 Minimum Norm Solution and the Moore–Penrose Pseudoinverse

Idea: Among the infinitely many solutions we pick the one with *smallest* norm, i.e.,

$$\min_{x \in \hat{S}} \|x\|_2^2 \quad (\hat{S} := \{x \in \mathbb{R}^n : A^T A x = A^T b\}). \quad (16)$$

→ We enforce uniqueness by determining a specific selection criterion.

#### Characterization of the minimum-norm solution

$$x^+ := \arg \min_{x \in \hat{S}} \|x\|_2^2$$

**Theorem 6.5 (Minimum-Norm Least Squares)** Let  $A \in \mathbb{R}^{m \times n}$  and  $b \in \mathbb{R}^m$ . Then

$$x^+ = \lim_{\delta \rightarrow 0} x_\delta$$

solves the minimum-norm least squares problem (16). Here,  $x_\delta = (A^T A + \delta I)^{-1} A^T b$  is the solution of the regularized least squares problem (14).

*Proof:* Uses the singular value decomposition.

## Remarks

## The Moore–Penrose Pseudoinverse

Let us explain why the term *pseudoinverse* is used here.

Let  $A \in \mathbb{R}^{m \times n}$  with  $m \neq n$ , then  $A$  and the corresponding function  $f_A: \mathbb{R}^n \rightarrow \mathbb{R}^m$  can't be invertible. In fact, there can't be a one-to-one relation between the spaces  $\mathbb{R}^n$  and  $\mathbb{R}^m$  in this case.

- The two spaces have different dimensions. For instance, a single nonzero vector can explain a line ( $\mathbb{R}$ ), but two independent vectors are needed to explain a plane ( $\mathbb{R}^2$ ).
- One could say that  $\mathbb{R}^n$  and  $\mathbb{R}^m$  are “differently large” if  $m \neq n$ .

**However:** We still aim at solving systems  $Ax = b$  for  $A \in \mathbb{R}^{m \times n}$  with possibly  $m \neq n$ .

*Recall:* If  $A$  is invertible (then in particular  $m = n$ ), then  $x = A^{-1}b$  is the unique solution. The inverse is a function which maps the right-hand side to the unique solution.

- As seen above, the concept of an inverse matrix fails if  $m \neq n$ .

We finally show

- i) The limiting matrix is the Moore–Penrose Pseudoinverse:

$$A^+ := \lim_{\delta \rightarrow 0} (A^T A + \delta I)^{-1} A^T = V \Sigma^+ U^\top$$

- ii) Applying the Moore–Penrose Pseudoinverse to  $b$  gives the minimum-norm least squares solution:

$$x^+ = V \Sigma^+ U^\top b.$$

**To ii)** 1. Let us start with the simple case:  $A \in \mathbb{R}^{m \times n}$  diagonal

2. Now we use these ideas for the general case:  $A \in \mathbb{R}^{m \times n}$

## 6.5 Small Tour: Inverse Problems in Imaging

→ presented in an ipython notebook.

## **Vector Spaces**

# 7 Vector Spaces

## 7.1 Introduction

Preliminary remarks:

**Definition 7.1 (Group)** Let  $G$  be a set and  $* : G \times G \rightarrow G$  a function, so that

**G1** *Associativity:*  $\forall g_1, g_2, g_3 \in G : g_1 * (g_2 * g_3) = (g_1 * g_2) * g_3$

**G2** *Neutral element:*  $\exists e \in G \ \forall g \in G : g * e = g$

**G3** *Inverse element:*  $\forall g \in G \ \exists g^{-1} \in G : g * g^{-1} = e$

Then  $(G, *)$  is called **group**.

If in addition

**G4** *Commutativity:*  $\forall g_1, g_2 \in G : g_1 * g_2 = g_2 * g_1$ ,

then it is called a **commutative/abelian group**.

We now add more structure by abstracting the familiar properties of real numbers with summation (subtraction) and multiplication (division).

**Definition 7.3 (Field)** Let  $F$  be a set and  $+ : F \times F \rightarrow F$  and  $\cdot : F \times F \rightarrow F$  two functions such that

- F1  $(F, +)$  is a commutative group (with neutral element 0)
- F2  $(F \setminus \{0\}, \cdot)$  is a commutative group (with neutral element 1)
- F3 Distributivity (compatibility of  $+$  and  $\cdot$ )

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

$$(a + b) \cdot c = a \cdot c + b \cdot c$$

We now abstract the notion of vectors in  $\mathbb{R}^n$  and their properties.

**Definition 7.5 (Vector space)** Let  $\mathbb{F}$  be a field. A set  $V$  together with a function  $+$  (sum) and a function  $\cdot$  (scalar multiplication) with

$$\begin{aligned} + : V \times V &\rightarrow V & \cdot : \mathbb{F} \times V &\rightarrow V \\ (v, w) &\mapsto v + w & (\lambda, v) &\mapsto \lambda \cdot v \end{aligned}$$

is called **vector space (or linear space)** over  $\mathbb{F}$ , if the following axioms **VR1** and **VR2** hold:

**VR1**  $(V, +)$  is a commutative group with neutral element 0.

**VR2** The scalar multiplication is compatible with  $(V, +)$  in the following way:  
for  $\lambda, \mu \in \mathbb{F}, v, w \in V$  it holds that

- i)  $(\lambda + \mu) \cdot v = \lambda \cdot v + \mu \cdot v$
- ii)  $\lambda \cdot (v + w) = \lambda \cdot v + \lambda \cdot w$
- iii)  $\lambda \cdot (\mu \cdot v) = (\lambda \cdot \mu) \cdot v$
- iv)  $1 \cdot v = v$

Remarks:

- The vector space axioms allow for an abstract study and serve as an interface to the developed theory.  
For example, this is important for the study of linear equations, where the sought after solutions are functions (e.g., differential or integral equations). We then look for solutions in particular function spaces (typically infinite-dimensional which we approximate with finite-dimensional ones on the computer).
- Often one equips vector spaces with additional structure: Norm (abstract notion of length), inner product (relates to angles and orthogonality), topology (relates to limits, continuity and connectedness),...  
→ Below we will introduce inner product spaces (a preliminary stage to so-called Hilbert Spaces)







## 7.2 Revisit: Linear Combination, Linear Independence, Basis

Based on the more abstract functions  $+$  and  $\cdot$  we can generally define:

**Definition 7.7** Let  $V$  be a vector space over the field  $\mathbb{F}$  and  $v_1, \dots, v_r \in V$ . Then, we define

a) **Linear combination:**

$$\sum_{j=1}^r \lambda_j v_j \in V, \quad \lambda_j \in \mathbb{F}$$

b) **Span (set of all linear combinations):**

$$\text{span}(v_1, \dots, v_r) := \left\{ \sum_{j=1}^r \lambda_j v_j : \lambda_j \in \mathbb{F}, j = 1, \dots, r \right\} \in V$$

d) The vectors  $v_1, \dots, v_r$  are called **linearly independent**, if

$$\sum_{j=1}^r \lambda_j v_j = 0 \Rightarrow \lambda_j = 0, \forall j = 1, \dots, r$$

e) The vectors  $v_1, \dots, v_r$  are called **basis of  $V$** , if

- i)  $v_1, \dots, v_r$  are linearly independent,
- ii)  $\text{span}(v_1, \dots, v_r) = V$ .

With the same proof as for  $\mathbb{F}^n$  we can show:

**Corollary 7.8** For vectors  $v_1, \dots, v_r \in V$  the following statements are equivalent:

- i)  $v_1, \dots, v_r$  are linearly independent
- ii) every vector  $v \in \text{span}(v_1, \dots, v_r)$  can be uniquely linearly combined from the set  $\{v_1, \dots, v_r\}$ .
- iii) None of  $v_i$  for  $i = 1, \dots, r$  can be written as a linear combination of the other.

Remark: Note that these notions coincide with the ones from the linear algebra section. However, now "vectors" are elements from any vector space  $V$  and could thus be vectors in the discrete sense, or matrices, or functions,...

### **Example 7.9**

## 7.3 Linear functions: kernel, image, matrix representation

**Definition 7.10 (Linear function)** Let  $V, W$  be two vector spaces over  $\mathbb{F}$ . Then a function  $f : V \rightarrow W$  is called an  $\mathbb{F}$ -linear function, if

$$f(\lambda v_1 +_V v_2) = \lambda f(v_1) +_W f(v_2) \quad \forall v_1, v_2 \in V, \lambda \in \mathbb{F}.$$

The set of all linear functions is denoted by  $\text{Hom}_{\mathbb{F}}(V, W)$  (homomorphisms).

For  $f \in \text{Hom}_{\mathbb{F}}(V, W)$  we define the **kernel**  $\ker(f)$  and the **image**  $\text{Im}(f)$  as

$$\ker(f) := \{v \in V : f(v) = 0\} \subset V,$$

$$\text{Im}(f) := \{f(v) \in W : v \in V\} \subset W.$$

## Matrix Representation of Linear Mappings

Next we show that for finite-dimensional vector spaces  $V$  and  $W$ , say  $\dim(V) = n$  and  $\dim(W) = m$ , the set of all linear functions from  $V$  to  $W$  is equivalent to the set of all  $m \times n$  matrices.



### Remark 7.12

- The matrix  $\mathcal{M}_{\{w_1, \dots, w_m\}}^{\{v_1, \dots, v_n\}}(f)$  is sometimes called *transformation matrix* (in German: *Darstellungsmatrix*).
- It is the matrix representation of  $f$  under the given bases and it tells us how to transform the coordinates!
- Also, it establishes a one-to-one relation between  $\text{Hom}_{\mathbb{F}}(V, W)$  and  $\mathbb{F}^{m \times n}$ , i.e.,

$$\text{Hom}_{\mathbb{F}}(V, W) \simeq \mathbb{F}^{m \times n}.$$





## 7.4 Inner Product Spaces

We now add more structure to a vector space: We equip it with an inner product.

**Definition 7.14** Let  $V$  be a vector space over  $\mathbb{F}$ .

- A mapping  $(\cdot, \cdot): V \times V \rightarrow \mathbb{F}$  is called **inner product** on  $\mathbb{F}^n$  if it satisfies:
  - i) **Hermitian:**  $\forall x, y \in V : (x, y) = \overline{(y, x)}$ ,
  - ii) **Linear in its second argument:**  $\forall x, y_1, y_2 \in V, \lambda \in \mathbb{F} : (x, y_1 + \lambda y_2) = (x, y_1) + \lambda(x, y_2)$ ,
  - iii) **Positive definite:**  $\forall x \in V \setminus \{0\} : (x, x) > 0$ .
- We call  $(V, (\cdot, \cdot))$  an **Euclidean vector space or inner product space**.

For simplicity we now stick to  $V = \mathbb{R}^n$ . We can show the following characterization:

**Theorem 7.15** Let  $(\cdot, \cdot): \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ , then

$$(\cdot, \cdot) \text{ inner product on } \mathbb{R}^n \Leftrightarrow \exists A \in \mathbb{R}_{spd}^{n \times n} \forall x, y \in \mathbb{R}^n : (x, y) := (x, y)_A := (x, Ay)_2.$$

In words: There is a one-to-one relation between all inner products on  $\mathbb{R}^n$  and all symmetric and positive definite matrices in  $\mathbb{R}^{n \times n}$ . The proof is constructive and reveals that the entries of  $A$  are given by

$$a_{ij} := (e_i, e_j)$$

where  $e_j$  denote the standard basis vectors.



## The important relation to Data Science/Optimization

- The gradient (times  $(-1)$ ) of a function determines a descent direction. Moving in this direction will minimize the function
- The gradient is precisely the vector representing the derivative of a function.
- You will learn how to accelerate the gradient method by representing it in another inner product, e.g., the one induced by the Hessian  $A = H_f(x)$  (this will give you Newton type method!).

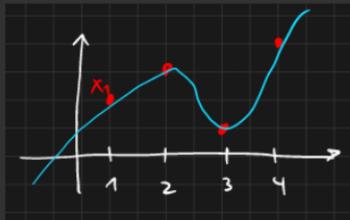
# **Calculus**

## 8 Nonlinear Aspects

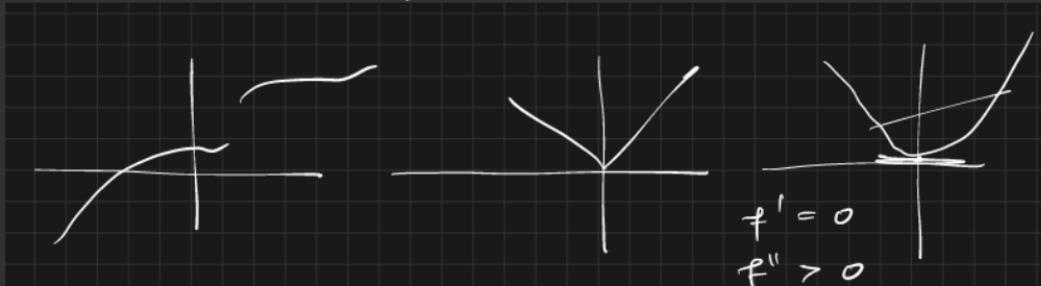
We will touch upon the following topics:

- continuous and differentiable functions
- partial derivatives, gradient, Jacobian
- (chain rule)
- In exercise: Taylor approximation and Newton's method

Until now we have worked with "discrete objects", say  $x \in \mathbb{R}^n$ ,  $\{1, \dots, n\} \rightarrow \mathbb{R}$ ,  $i \mapsto x_i$



Now, vectors become functions  $f : \mathbb{R} \rightarrow \mathbb{R}$



## 8.1 Motivation

Let us first recall the definition of a linear function. Consider a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ , then

$$f \text{ linear } \stackrel{\text{Def}}{\Leftrightarrow} \forall x, y \in \mathbb{R}^n, \lambda \in \mathbb{R} : f(\lambda \cdot x + y) = \lambda \cdot f(x) + f(y).$$

The prototype of a linear function between finite dimensional spaces is the matrix–vector product, more precisely,

$$A \in \mathbb{R}^{m \times n}, f_A(x) := Ax.$$

We say  $f$  is **nonlinear**, if it is not linear.

Nonlinear function may extend our modeling choice significantly and may help to explain complicated relations, such as

$$\begin{array}{ccc} z_i & \rightarrow & y_i \\ \in \mathbb{R}^p & & \in \mathbb{R}^q, i = 1, \dots, m. \\ [\text{image}] & & [\text{feature}] \end{array}$$

Until now, we have consider models with *linear* dependency of the parameters:

$$f_x(z) = \sum_{k=1}^n x_k \cdot f_k(z) \approx y.$$

We determined the parameters  $x = (x_k)_k$  by solving a (potentially regularized) least squares problem of the form

$$\min_x L(x; (z_i, y_i)) + R(x), \quad \left( \text{e.g., Ridge Regression } R(x) := \frac{\delta}{2} \|x\|_2^2 \right),$$

where the cost function has the form

$$\sum_{i=1}^m \|f_x(z_i) - y_i\|_2^2 = \|A_z x - y\|_2^2 =: L(x, (z_i, y_i)).$$

The specialty of this kind of minimization problem is that we can solve it via the normal equation, which is a *linear* equation.

Now let us consider a **nonlinear model** (e.g. Neural Network); more precisely, nonlinear with respect to the sought-after parameters. More specifically, let us for example consider a model of the form

$$f_x(z) = (f_M \circ \dots \circ f_1)(z) = f_M(f_{M-1}(f \dots (f_1(z)) \dots ))$$

where the building blocks  $f_k$ , also called **layers**, are given by

$$f_k: \mathbb{R}^p \rightarrow [0, +\infty)^q, \quad f_k(z) := (A_k z + b_k)_+ \quad (\text{applied element-wise}),$$

with

$$\mathbb{R} \rightarrow [0, +\infty), \quad w_+ := \begin{cases} 0 : w < 0 \\ w : \text{else} \end{cases}$$

being the so-called **ReLU function** (Rectified Linear Unit), an example of a so-called **activation function**.

The matrices  $A_k \in \mathbb{R}^{q \times p}$  and vectors  $b_k \in \mathbb{R}^q$  are the parameters (also called **weights**) that need to be determined. If  $A_k$  is dense, the function  $f_k$  is called **fully connected layer** and if, e.g.,  $A_k$  is Toeplitz, then  $f_k$  is called **convolutional layer**.

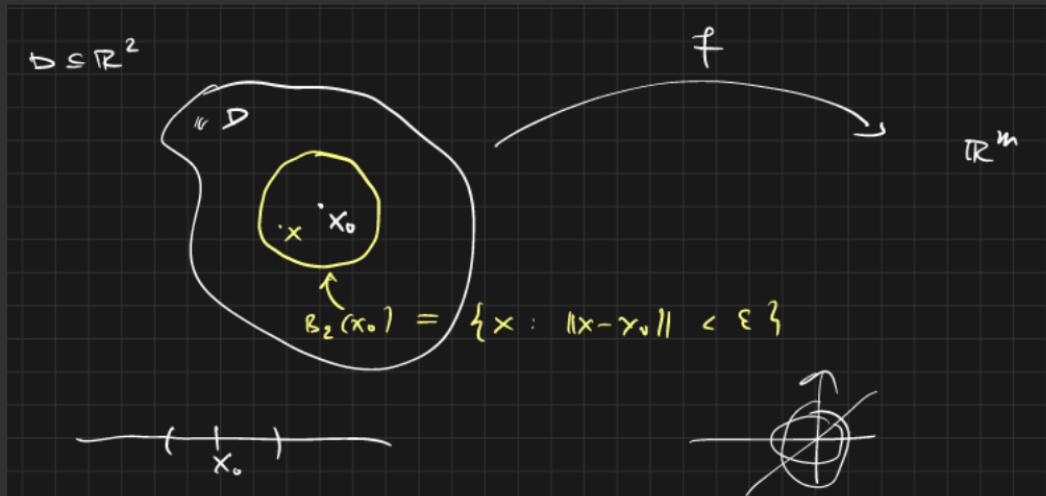
Due to the ReLU function  $(\cdot)_+$  the concatenated model  $f_x$  is highly nonlinear.

Similarly to the linear case, we aim to find suitable parameters/weights  $x := (A_k, b_k)_k$  that best describe the model with respect to a certain cost function:

$$\min_{x:=(A_k, b_k)_k} L(x; (z_i, y_i)) + R(x) =: F(x) \quad (\leftarrow F \text{ highly nonlinear})$$

Before we continue with some standard definitions from calculus, a preliminary remark:

The concepts of continuity and differentiability in the context of a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  are "local" concepts, i.e., they are required to hold in a small neighborhood of a point  $x_0 \in \mathbb{R}^n$ .



## 8.2 Continuity and Differentiability

In the following we consider neighborhoods of the form  $B_\varepsilon(x_0) := \{x \in \mathbb{R}^n : \|x - x_0\| < \varepsilon\}$ .

**Definition 8.1 (Continuous and differentiable function)**

Let  $D \subseteq \mathbb{R}^n$ ,  $f : D \rightarrow \mathbb{R}^m$  and  $x_0 \in D$  with  $B_\varepsilon(x_0) \subseteq D$  for some  $\varepsilon > 0$ . Then

i)  $f$  is called **continuous** at  $x_0$ , if

$$\lim_{n \rightarrow 0} \|f(x_n) - f(x_0)\|_2 = 0$$

for all sequences  $(x_n)_{n \in \mathbb{N}} \subseteq B_\varepsilon(x_0)$  for which  $x_n \rightarrow x_0$ .

ii)  $f$  is called **differentiable** at  $x_0$ , if there is a linear mapping  $A : \mathbb{R}^n \rightarrow \mathbb{R}^m$  such that

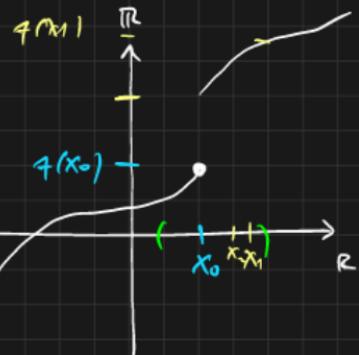
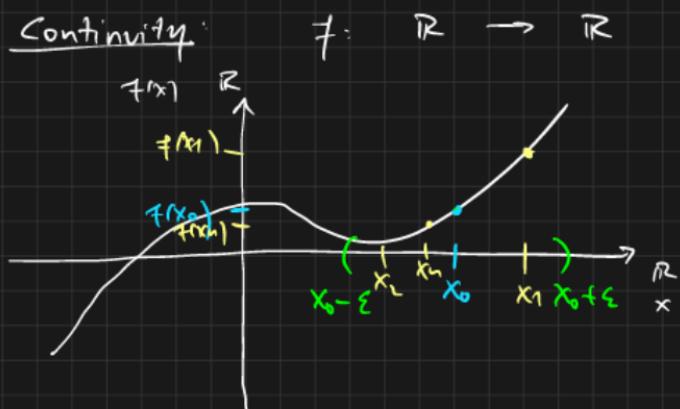
$$\lim_{n \rightarrow \infty} \frac{\|(f(x_0) + Ah_n) - f(x_0 + h_n)\|}{\|h_n\|} = 0$$

for all sequences  $(h_n)_n$  with  $x_0 + h_n \subseteq B_\varepsilon(x_0)$ ,  $\lim_{n \rightarrow \infty} \|h_n\| \rightarrow 0$ .

Since the linear function  $A$  depends on  $f$  and  $x_0$ , we denote it as  $Df(x_0) := A$  and call it (Fréchet) derivative.

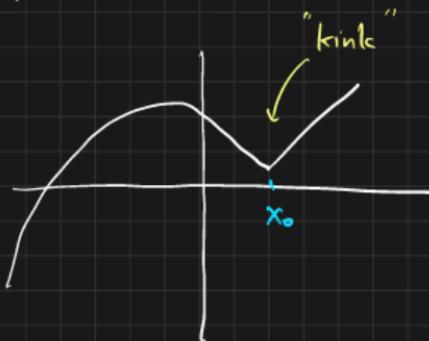
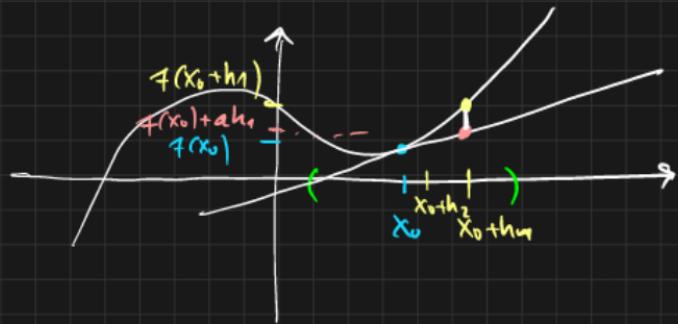
If  $f$  is continuous/differentiable at any point  $x_0 \in D$ , we call  $f$  simply continuous/differentiable.

Continuity



Differentiable

$f: \mathbb{R} \rightarrow \mathbb{R}$ ,  $Df(x_0)$ ,  $\frac{\epsilon}{|h|} |f(x_0 + a \cdot h) - f(x_0)|$



## Examples: Continuity

i)  $f : \mathbb{R} \rightarrow \mathbb{R}, x \mapsto |x| = \begin{cases} x & x \geq 0 \\ -x & x < 0 \end{cases}$

Let  $x_0 \in \mathbb{R}, (x_n)_{n \in \mathbb{N}}, x_n \xrightarrow{n \rightarrow \infty} x_0$ , then

$$0 \leq |f(x_n) - f(x_0)| = ||x_n| - |x_0|| \leq |x_n - x_0| \xrightarrow{n \rightarrow 0} 0$$

$\Rightarrow f$  is continuous.

ii)  $f : \mathbb{R} \rightarrow \mathbb{R}, x \mapsto x^2$

Let  $x_0 \in \mathbb{R}, x_n \rightarrow x_0$ , then

$$|f(x_n) - f(x_0)| = |x_n^2 - x_0^2| = |(x_n - x_0)(x_n + x_0)| = \underbrace{|x_n - x_0|}_{\rightarrow 0} \underbrace{|x_n + x_0|}_{\rightarrow 2x_0} \xrightarrow{n \rightarrow \infty} 0$$

$\Rightarrow f$  is continuous.

iii)  $f : \mathbb{R} \rightarrow \mathbb{R}, f(x) := \begin{cases} 1 & x > 0 \\ -1 & x \leq 0 \end{cases}$

Let  $x_0 = 0, x_n \rightarrow 0^+$ , then

$$|f(x_n) - f(x_0)| = |1 - (-1)| = 2 \not\rightarrow 0$$

$\Rightarrow f$  is not continuous.

## Examples: Differentiability

i)  $f : \mathbb{R} \rightarrow \mathbb{R}, x \mapsto ax, a \in \mathbb{R}$

Let us consider the surrogate  $Df(x_0)(h) := ah$  and a sequence  $h_n \rightarrow 0$ . Then

$$\frac{1}{|h_n|} |f(x_0) + Df(x_0)h_n - f(x_0 + h_n)| = \frac{1}{|h_n|} |ax_0 + ah_n - a(x_0 + h_n)| = 0 \xrightarrow{n \rightarrow \infty} 0$$

$\Rightarrow f$  is differentiable.

ii)  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m, x \mapsto Ax, A \in \mathbb{R}^{m \times n}$

Let us consider the surrogate  $Df(x_0)(h) := Ah$  and a sequence  $h_n \rightarrow 0$ . Then

$$\frac{1}{|h_n|} |f(x_0) + Df(x_0)h_n - f(x_0 + h_n)| = \frac{1}{|h_n|} |Ax_0 + Ah_n - A(x_0 + h_n)| = 0 \xrightarrow{n \rightarrow \infty} 0$$

$\Rightarrow f$  is differentiable.

iii)  $f : \mathbb{R} \rightarrow \mathbb{R}, x \mapsto |x|$

$f$  is **not** differentiable at  $x_0 = 0$ .

**Remark:** How can we identify continuous/differentiable functions?

- Many elementary functions (polynomials, trigonometric functions, exponential function,...) and operations ("+", "·", ...) to combine such elementary functions are continuous/differentiable.
- The concatenation of such functions is also continuous/differentiable!
- Examples:

- monomial  $x^k$  and polynomial (=linear combination)  $p(x) = \sum_{j=0}^m a_j x^j$
- exponential function  $e^x$  and sine function  $\sin(x) = \frac{1}{2i}(e^{ix} - e^{-ix})$

We will show in the exercise that differentiability is a stronger requirement than continuity:

**Theorem 8.2** *Every differentiable function is also continuous.*

Next, we introduce the directional derivative which often serves as a good starting point to find the (Fréchet) derivative of a function (especially in complex and confusing situations):

**Definition 8.3 (Directional derivative)** We assume that  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  is (Fréchet-) differentiable at  $x_0 \in \mathbb{R}^n$  with derivative  $Df(x_0)$ . For a  $v \in \mathbb{R}^n$ , the limit

$$Gf(x_0)(v) := \lim_{t \rightarrow 0^+} \frac{f(x_0 + tv) - f(x_0)}{t}$$

exists and it concides with the Fréchet derivative, i.e.,  $Gf(x_0)(v) = Df(x_0)(v)$ .

We call  $Gf(x_0)(v)$  the **directional derivative** at  $x_0$  in the direction  $v$ . (Gâteaux derivative)

**Remark:**

The Gâteaux derivative may exist, even if  $f$  is not Fréchet differentiable (e.g.  $x \mapsto |x|$ ,  $x_0 = 0$ ).

## Examples:

i)  $f : \mathbb{R} \rightarrow \mathbb{R}$ ,  $x \mapsto |x|$ ,  $x_0 = 0$  (not Fréchet-differentiable)

$$\begin{aligned} \text{a)} \ v \geq 0: \ Gf(x_0)(v) &= \lim_{t \rightarrow 0^+} \frac{1}{t}(f(x_0 + tv) - f(x_0)) = \lim_{t \rightarrow 0^+} \frac{1}{t}(tv) = 1 \cdot v \\ \text{b)} \ v < 0: \ Gf(x_0)(v) &= \lim_{t \rightarrow 0^+} \underbrace{\frac{1}{t}(f(x_0 + tv) - f(x_0))}_{=-tv} = (-1) \cdot v \end{aligned}$$

ii)  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $x \mapsto \|x\|_2^2 = x^T x$ ,  $v \in \mathbb{R}^n$ ,  $x_0 \in \mathbb{R}^n$

$$\begin{aligned} Gf(x_0)(v) &= \lim_{t \rightarrow 0^+} \frac{f(x_0 + tv) - f(x_0)}{t} \\ &= \lim_{t \rightarrow 0^+} \left( \frac{(x_0 + tv)^T (x_0 + tv)}{x_0^T x_0 + 2tx_0^T v + t^2 v^T v} - x_0^T x_0 \right) \frac{1}{t} \\ &= (2x_0)^T v \end{aligned}$$

Consider  $v = \sum_{j=1}^n v_j e_j$ , where  $e_1, \dots, e_n$  denote the standard basis in  $\mathbb{R}^n$ , then

$$Df(x_0)(v) = \sum_{j=1}^n v_j \underbrace{Df(x_0)(e_j)}_{\substack{\mathbb{R}^n \rightarrow \mathbb{R}^m \\ \in \mathbb{R}^m}}$$

**Definition 8.4 (Partial derivative)** Let  $f : D \rightarrow \mathbb{R}^m$ ,  $D \subseteq \mathbb{R}^n$  be (Fréchet-)differentiable in  $x_0 \in D$ . We define the so-called **partial derivatives** of  $f$  at  $x_0$  with respect to the  $j$ -th variable by:

$$\frac{\partial}{\partial x_j} f(x_0) := Df(x_0)(e_j),$$

where  $e_j$  is the  $j$ -th standard basis vector.

Now again with  $v = \sum_{j=1}^n v_j e_j$  we find

$$\begin{aligned} Df(x_0)(v) &= \sum_{j=1}^n v_j Df(x_0)(e_j) \\ &= \begin{pmatrix} | & & | \\ Df(x_0)(e_1) & \cdots & Df(x_0)(e_n) \\ | & & | \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \\ &= \begin{pmatrix} | \\ \frac{\partial}{\partial x_1} f(x_0) \\ | & \cdots & | \\ \frac{\partial}{\partial x_n} f(x_0) \end{pmatrix} v \\ &= \underbrace{J_f(x_0)}_{\in \mathbb{R}^{m \times n}} \cdot v \\ f : \mathbb{R}^n &\rightarrow \mathbb{R}^m, f(x) = \begin{pmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{pmatrix}, f_i : \mathbb{R}^n \rightarrow \mathbb{R} \end{aligned}$$

Since  $Df(x_0) : \mathbb{R}^n \rightarrow \mathbb{R}^m$  is linear it can be represented by a matrix:

**Lemma 8.5 (Jacobian)** Let  $f : \mathbb{R}^n \supset D \rightarrow \mathbb{R}^m$  be differentiable at  $x_0 \in D$  with derivative  $Df(x_0) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ . Then the so-called **Jacobian matrix**

$$J_f(x_0) := \mathcal{M}_I^I(Df(x_0)) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x_0) & \cdots & \frac{\partial f_1}{\partial x_n}(x_0) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(x_0) & \cdots & \frac{\partial f_m}{\partial x_n}(x_0) \end{pmatrix} \in \mathbb{R}^{m \times n}$$

is the matrix representation of  $Df(x_0)$  with respect to the standard bases in  $\mathbb{R}^n$  and  $\mathbb{R}^m$ .

In the special case, that the Jacobian matrix is just one row we give it a special name:

**Definition 8.6 (Gradient)** Let  $f : \mathbb{R}^n \supset D \rightarrow \mathbb{R}$  be differentiable at  $x_0 \in D$ , then

$$J_f(x_0)^T = \begin{pmatrix} \frac{\partial f}{\partial x_1}(x_0) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x_0) \end{pmatrix} =: \nabla f(x_0)$$

is called the **gradient of  $f$**  at  $x_0 \in D$ .

## Example

## 8.3 Solving Nonlinear Equations: Taylor Approximation and Newton's Method

The next result is on the approximation quality of the derivative:

**Lemma 8.7 (Taylor approximation)** Let  $f : \mathbb{R}^n \supset B_\varepsilon(\hat{x}) \rightarrow \mathbb{R}^n$  be differentiable at  $\hat{x}$  with some  $\varepsilon > 0$ . Assume further that there is a (Lipschitz) constant  $L \geq 0$  such that the Jacobian  $J_f$  satisfies

$$\|J_f(y) - J_f(x)\| \leq L\|y - x\|, \quad \forall x, y \in B_\varepsilon(\hat{x}). \quad (18)$$

Then, there holds

$$\|f(y) - [f(x) + J_f(x)(y - x)]\| \leq \frac{L}{2}\|y - x\|^2, \quad \forall x, y \in B_\varepsilon(\hat{x})$$

which we rephrase with the notation:

$$f(y) = f(x) + J_f(x)(y - x) + \mathcal{O}(\|y - x\|^2).$$

Let us apply Taylor approximation to solve nonlinear systems: The idea is to locally approximate the nonlinear function by its linear derivative and then solve many linear systems.

- Situation: Consider for a potentially nonlinear function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$  and the nonlinear system  $f(\hat{x}) = 0$
- Aim: Determine the solution  $\hat{x}$  (iteratively/numerically)
- Idea: Define an iterative scheme  $x^{k+1} := x^k + \Delta x^k$  where the increment is derived as follows:

$$\begin{aligned} 0 &\stackrel{!}{=} f(x^{k+1}) = f(x^k + \Delta x^k) \approx f(x^k) + J_f(x^k)\Delta x^k \quad (\leadsto \text{solve for } \Delta x^k) \\ \Leftrightarrow J_f(x^k) \cdot \Delta x^k &= -f(x^k) \quad (\text{linear equation}) \\ \Leftrightarrow \Delta x^k &= -J_f(x^k)^{-1}f(x^k) \quad (\text{invertibility of the derivative at each } x_k \text{ assumed!}) \\ x^k &\rightarrow \hat{x} \end{aligned}$$

One can show the following convergence result of this approach:

**Theorem 8.8 (simplified Newton-Kantorovich)** Let  $f : \mathbb{R}^n \supset B_\varepsilon(\hat{x}) \rightarrow \mathbb{R}^n$  be differentiable with invertible derivative for some  $\varepsilon > 0$  and  $f(\hat{x}) = 0$ . Assume the Lipschitz condition (18) and the existence of an upper bound  $\|J_f(x)^{-1}\| < M$  for some  $M < \infty$  and for all  $x \in B_\varepsilon(\hat{x})$ . Then, the Newton iteration

$$x^{k+1} := x^k + \Delta x^k, \quad \text{where } \Delta x^k \text{ solves } f(x^k) + J_f(x^k)\Delta x^k = 0$$

converges quadratically to  $\hat{x}$ , provided  $x^1$  is chosen sufficiently close to  $\hat{x}$ , i.e.

$$\|x^{k+1} - \hat{x}\| \leq c\|x^k - \hat{x}\|^2, \quad c < \infty.$$

## Remark

In many cases, Newton's method does not work right out of the box, because the starting vector  $x^1$  is too far away from the solution. Then, techniques for adaptive step-length reduction (damping, relaxation, line-search) have to be used in order to enforce convergence. Details of these approaches fill multiple books. When Newton's method works, i.e., after an initial damped phase, it gets super fast.

### Take-away messages:

- Derivatives → local linear approximation to the function
- Newton's method → solves nonlinear systems by solving many linear problems in each step

## 8.4 The Chain Rule and Back Propagation

The chain rule lies at the heart of back propagation. It tells us how to compute the derivative of concatenated functions:

**Theorem 8.9 (Chain rule)** Consider mappings  $g : \mathbb{R}^\ell \supset D_g \rightarrow D_f \subset \mathbb{R}^m$  differentiable in  $x_0 \in D_g$  with Jacobian  $J_g(x_0)$  and  $f : \mathbb{R}^m \supset D_f \rightarrow \mathbb{R}^n$ , differentiable in  $g(x_0) \in D_f$  with Jacobian  $J_f(g(x_0))$ . Then, the concatenation is differentiable with Jacobian  $J_{f \circ g}(x_0)$  and

$$D(f \circ g)(x_0) = Df(g(x_0)) \circ Dg(x_0) \quad \text{and} \quad J_{f \circ g}(x_0) = J_f(g(x_0)) \cdot J_g(x_0).$$

**Example 8.10** Let us revisit our regularizer from the imaging example:

Consider  $D \in \mathbb{R}^{p \times n}$  and the linear function  $g : \mathbb{R}^n \rightarrow \mathbb{R}^p$ ,  $g(x) := Dx$ . Then for all  $x \in \mathbb{R}^n$  we easily find

$$J_g(x) = D.$$

Also, let  $f : \mathbb{R}^p \rightarrow \mathbb{R}$ ,  $f(y) := \frac{1}{2}y^\top y = \frac{1}{2}\|y\|_2^2$ , then we have seen above that, for all  $y \in \mathbb{R}^p$ ,

$$J_f(y)^\top = \nabla f(y) = \frac{1}{2}2y = y.$$

Then the concatenation  $h := (f \circ g) : \mathbb{R}^n \rightarrow \mathbb{R}$  is given by

$$h(x) = \frac{1}{2}\|Dx\|_2^2$$

with gradient, at  $x \in \mathbb{R}^n$ , obtained from the chain rule

$$\nabla h(x) = J_h(x)^\top = (J_f(g(x)) \cdot J_g(x))^\top = D^\top \nabla f(g(x)) = D^\top g(x) = D^\top Dx.$$