Advanced Numerical Analysis

Lecture 1
Spring 2025

EPFL

Daniel Kressner

What is numerical analysis?

- *Approximate* solution of mathematical problems that are difficult or impossible to solve by hand
- Design and analysis of algorithms
- Analysis and control of error (approximation, floating point, measurement)
- Beautiful mathematics[1]

[1] sometimes 😃

What is numerical analysis?

- Daniel Kressner, Lecturer
  - Professor of mathematics at MATH, EPFL
  - Research areas: numerical linear algebra/analysis, design and fast implementation of algorithms, applications (e.g., simulating plasma in EPFL's tokamak)
- Fabio Matti, principal assistant
  - PhD student of mathematics at EDMA
  - Research topic: Randomized numerical linear algebra
- Team of assistants: Thomas Michel, Peter Oehme, Marija Vukšić, Zhipeng Xue
- Ways to contact us: lectures/exercise sessions, *Ed Discussion Board*, e-mail

Lectures

- Thursday 14h15–15h00, 15h15–16h00, CE 1 5
- First lecture 20.2., last lecture 22.5.
- Lecture notes on moodle will be provided chapter-by-chapter during the semester (Chapter 1 is already online).
- Additional references are provided, but only material discussed in class room, the lecture notes, and the exercises is relevant.
- Lectures mainly on blackboard. Slides mainly for illustration (no extra material).

Exercises

- Exercise session: Friday 10h15–12h00
  CO 020 (last names A–F) and CO 021 (last names G–Z)
- First session 21.2., last sesssion 30.5.
- Each week there will be an exercise sheet.
- In week n, the exercise sheet will be put online on Wednesday and contains 3 parts:
  1. Quiz:
     Concerns the understanding of lecture material, will be discussed in the beginning of the lecture of week $n + 1$
  2. Exercises:
     To be solved during (and after) the exercise session on Friday in week n. Several assistants will be available to help you.
     Corrections are put online in week $n + 1$.
  3. **Homework:**
     Mix of theoretical and programming exercises to be submitted until Friday 10h15 of week $n + 1$. Homework grading will be handled by Peter Oehme.

Final exam

- Closed book exam.
  One *handwritten* A4 page (both sides) allowed
- Only topics discussed in lecture and exercises relevant for the exam (detailed summary during the last week).
- Basic knowledge of Python needed to solve the exam.
  This is not a programming course!
- Homework will contribute to the grade as follows:
  - Each week you can attain between 0 and 2 points.
  - $Y = \text{sum of your points}/(2 \cdot \text{number of homeworks})$
  - $X = \text{grade in exam}$, $X = 6$ is possible!
  - Final grade $= \min(X + Y, 6)$

Notes on Python

- This week's exercise session will be concerned with training lecture-relevant Python skills by going through tutorials and solving simple exercises.
- Use Jupyter Notebooks on a local installation or Noto:

  `https://go.epfl.ch/advanced-numerical-analysis-2025`
- Reach out to 1234 if you have installation problems.
- For Python related questions (beyond installation issues) use Ed Discussion Board.

Contents of this lecture

1. Representation of numbers
2. Numerical integration
3. Polynomial interpolation
4. Linear systems – small matrices
5. Linear systems – large matrices
6. Regression and least squares
7. Fourier transform

...and now a brief journey...

1. Representation of numbers

- Computers are finite
  $\rightsquigarrow$ no chance to operate with all numbers in $\mathbb{R}$, $\mathbb{Q}$, or even $\mathbb{N}$
- Floating point numbers $\mathbb{F}$ = well-chosen finite subset of $\mathbb{R}$
- $\mathbb{F}$ is not closed under $+, *, \ldots$! 😦

A US$370 million failure due to lack of closedness:

https://en.wikipedia.org/wiki/Ariane_flight_V88

2. Numerical integration

What s the integral of exp(-sqrt(sin(tan(x))) from 1 to 2?

The integral you're asking about is:

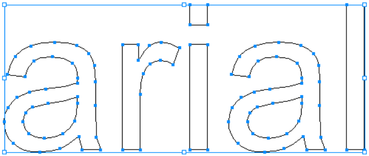
$$\int_1^2 \exp\left(-\sqrt{\sin(\tan(x))}\right)\, dx$$

This integral is quite complex due to the composition of functions inside the exponential and the square root. It does not have a straightforward analytical solution using elementary functions. Therefore, it typically requires numerical methods to evaluate.

- School/Analysis@EPFL: Approx of $\int_a^b$ by rectangles/trapezoids.
- Can one do better?

3. Polynomial interpolation

Modern font shapes are represented by piecewise polynomials:



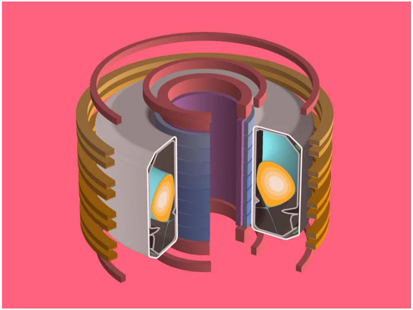
Polynomials need to pass through prescribed points + pieces need to glue smoothly together $\rightsquigarrow$ interpolation conditions.

See `https://jdhao.github.io/2018/11/27/font_shape_mathematics_bezier_curves/`
(The Mathematics behind Font Shapes — Bézier Curves and More)

4./5. Linear Systems

A simple equation at the heart of computational simulations:

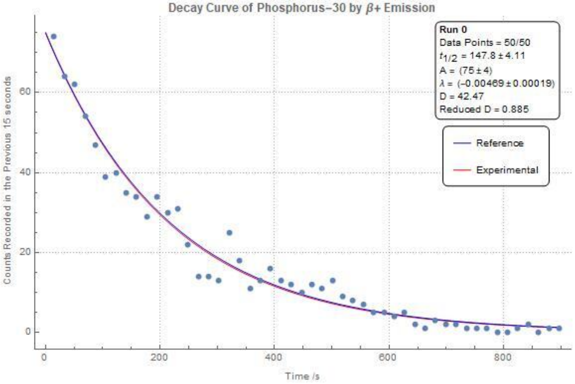
$$Ax = b$$



Collaboration between MATH + SPC:

Simulation of plasma in EPFL's tokamak requires solving linear systems with millions of unknowns!

6. Regression and least squares



How to fit an exponential curve to a data set of radioactive decays?

7. Fourier transforms

Fourier transforms are everywhere:

- Imaging: JPEG, Computed Tomography, filtering, ...
- Audio: MP3, filtering, ...
- Videos: Compression (MPEG-2, H.264 / AVC, VP9, H.265 / HEVC, ...)

The Fast Fourier Transform (FFT) runs the world!

> Fast Fourier transforms are widely used for applications in engineering, music, science, and mathematics. The basic ideas were popularized in 1965, but some algorithms had been derived as early as 1805.[1] In 1994, Gilbert Strang described the FFT as "the most important numerical algorithm of our lifetime",[3][4] and it was included in Top 10 Algorithms of 20th Century by the IEEE magazine *Computing in Science & Engineering*.[5]

From
`https://en.wikipedia.org/wiki/Fast_Fourier_transform`

Finite differences

Analysis 1: Let $f : \mathbb{R} \to \mathbb{R}$ be differentiable at $x \in \mathbb{R}$. Then

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

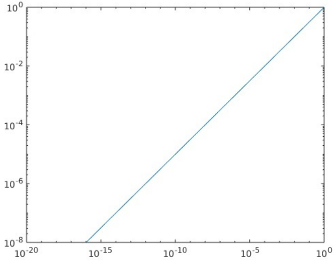
Expectation: Approximation of $f'(x)$ by finite difference quotient tends to get better as h approaches 0.

Python:

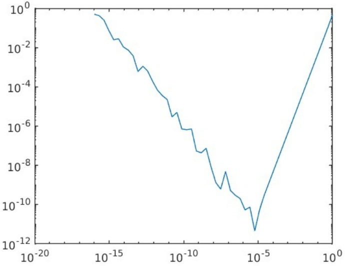
```
import numpy as np
import matplotlib.pyplot as plt

h = np.logspace(-16, 0)
fd = (np.exp(1 + h) - np.exp(1)) / h
error = np.abs(fd - np.exp(1))
plt.loglog(h, error)
```

Quiz: Which of these figures corresponds to the Python program?

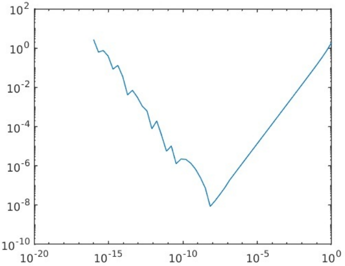
(A)



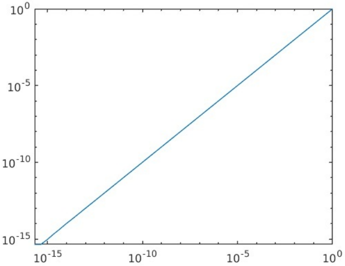
(B)



(C)



(D)



The number e (Example 1.1 in lecture notes)

Analysis 1:

$$e = \lim_{n\to\infty} (1 + \frac{1}{n})^n.$$

Expectation: $e_n = (1 + \frac{1}{n})^n$ becomes increasingly closer to e as n gets larger.

PYTHON

```
# Approximation of e, numpy package needed to include e
import numpy as np
for i in range(1,16):
    n = 10.0 ** i; en = (1 + 1/n) ** n
    print('10^%2d %20.15f %20.15f' % (i,en,en-np.e))
```

The number *e* (Example 1.1 in lecture notes)

n	Computed $\hat{e}_n$	Error $\hat{e}_n - e$
10^1	2.593742460100002	-0.124539368359044
10^2	2.704813829421529	-0.013467999037517
10^3	2.716923932235520	-0.001357896223525
10^4	2.718145926824356	-0.000135901634689
10^5	2.718268237197528	-0.000013591261517
10^6	2.718280469156428	-0.000001359302618
10^7	2.718281693980372	-0.000000134478673
10^8	2.718281786395798	-0.000000042063248
10^9	2.718282030814509	0.000000202355464
10^{10}	2.718282053234788	0.000000224775742
10^{11}	2.718282053357110	0.000000224898065
10^{12}	2.718523496037238	0.000241667578192
10^{13}	2.716110034086901	-0.002171794372145
10^{14}	2.716110034087023	-0.002171794372023
10^{15}	3.035035206549262	0.316753378090216

Quiz

Only one answer is correct!

Let $x \in \mathbb{R}$.

(A) If x has a finite decimal representation then x also has a finite binary representation.

(B) x has a finite binary representation if and only if x has a finite hexadecimal representation.

(C) If x has an infinite decimal representation then the representation of x in any base $\beta \geq 2$, $\beta \in \mathbb{N}$, is infinite.

(D) There always exists some $\beta \geq 2$, $\beta \in \mathbb{N}$, in which x has a finite representation.

Finite vs. infinite representations

Theorem 1.4.

Consider a nonzero rational number $x = \frac{p}{q}$, where $p, q \in \mathbb{Z}$ have no common divisor. Then x has a finite representation in base $\beta \geq 2$ if and only if each of the prime factors of the denominator q divides β.

Quiz

Only one answer is correct!

Let $\mathbb{F} = \mathbb{F}(10, 3, -3, 1)$.

(A) $23.4 \in \mathbb{F}$

(B) $3.141 \in \mathbb{F}$

(C) $-0.00732 \in \mathbb{F}$

(D) $10.0 \in \mathbb{F}$

IEEE 754 Standard

$\beta = 2$

Name	Size	Mantissa	Exponent	$x_{\min}$	$x_{\max}$
Single precision	32 bits	$23 + 1$ bit	8 bits	10^{-38}	10^{+38}
Double precision	64 bits	$52 + 1$ bit	11 bits	10^{-308}	10^{+308}

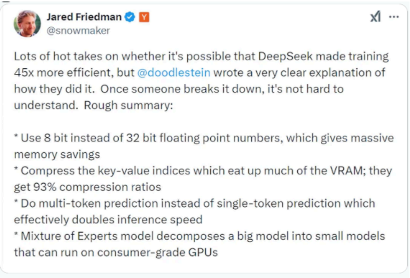
PYTHON

```
import sys
sys.float_info.min               # 2.2251e-308
sys.float_info.max               # 1.7977e+308
1 / 0                            # Divide by zero error
3 * float('inf')                 # inf
-1 / 0                           # Divide by zero error
0 / 0                            # Divide by zero error
float('inf') - float('inf')      # nan
```

Precisions for machine learning

$\beta = 2$

Name	Size	Mantissa	Exponent	$x_{\min}$	$x_{\max}$
bfloat16	16 bits	8 bits	8 bits	10^{-38}	10^{+38}
FP8	8 bits	4 bits	4 bits	10^{-2}	240


Jared Friedman
@snowmaker

Lots of hot takes on whether it's possible that DeepSeek made training 45x more efficient, but @doodlestein wrote a very clear explanation of how they did it. Once someone breaks it down, it's not hard to understand. Rough summary:

* Use 8 bit instead of 32 bit floating point numbers, which gives massive memory savings
* Compress the key-value indices which eat up much of the VRAM; they get 93% compression ratios
* Do multi-token prediction instead of single-token prediction which effectively doubles inference speed
* Mixture of Experts model decomposes a big model into small models that can run on consumer-grade GPUs

Quiz

1. What is the result of `x = -0.0; y = 1/x` in Python?

(A) `y = Inf`

(B) `y = -Inf`

(C) `y = NaN`

(D) `Error` 🤮

(E) It depends 😕