CS 328

Numerical Methods for Visual Computing and Machine Learning

Prof. Wenzel JAKOB

EPFL RGL Realistic Graphics Lab

Course structure

Thursday
CM13
10:15-12:00

Thurday
INF3, (BC07/08?)
16:15-18:00

Lecture:
- Concepts
- Live examples
- General Q&A

Exercise session:
- First 3 weeks: tutorials
- Homework Q&A

The TA team

PhD. TAs

Lovro Nuic

Ekrem Yilmazer

AEs

Johnny Kalajdzic

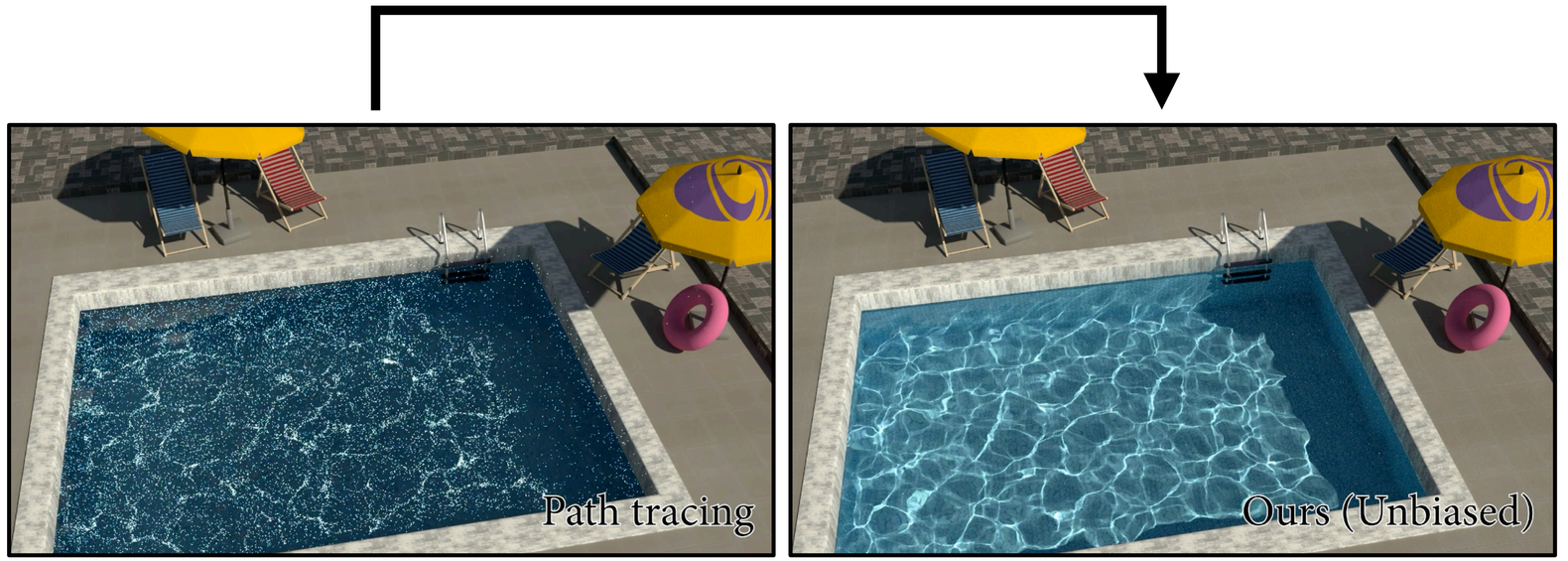
Ioana Jianu

Gaël Losada

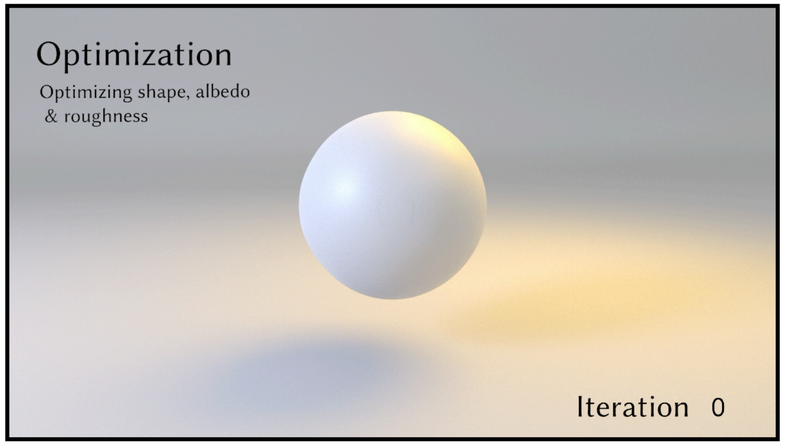
Pierre Pintado

About myself

- I lead the *Realistic Graphics Lab* (RGL) since 2016 (currently Associate Professor).
- My group does research on *physically-based rendering* and *inverse rendering*.
- The topics in this course are our daily bread.



[Zeltner et al. 2021]



[Vicini et al. 2022]

There isn't enough space

"We feel like sardines in a tincan."

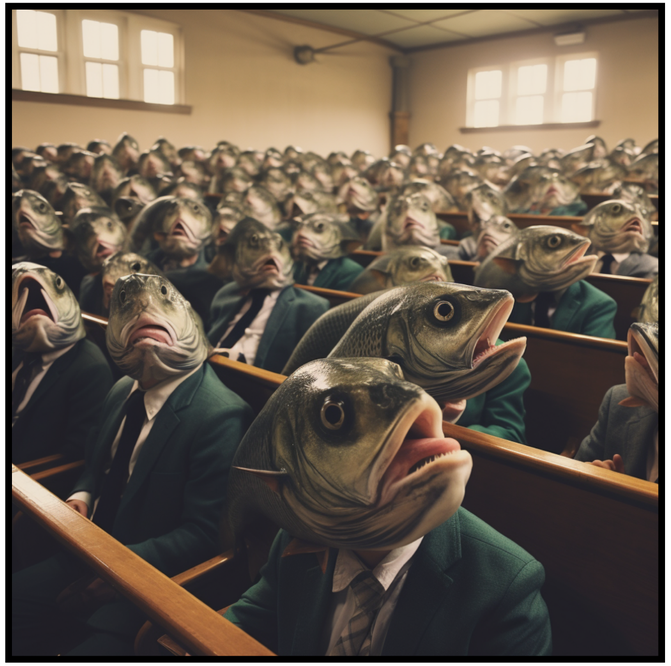
Significant growth in student numbers over the last years has led to a severe space issue on campus (will get *even worse* in future years).

Simultaneously: lots of construction happening on campus.

Also: conflict with SHS lectures for many of you.

Please don't blame me for these issues, they are not my fault.

Lecture recordings from 2023 are available.



Midjourney: *an incredibly packed lecture hall. The students are sardines.*

What is this course about?

.. what is a "Numerical Method", anyways?

Computer Graphics

Computer Vision

Machine Learning

Computational Photography

Weather simulation

Geometric Information Systems

...

Here be dragons

All of them have some kind of **numerical method** at their core

Number crunchers

Numerical Methods for Rendering

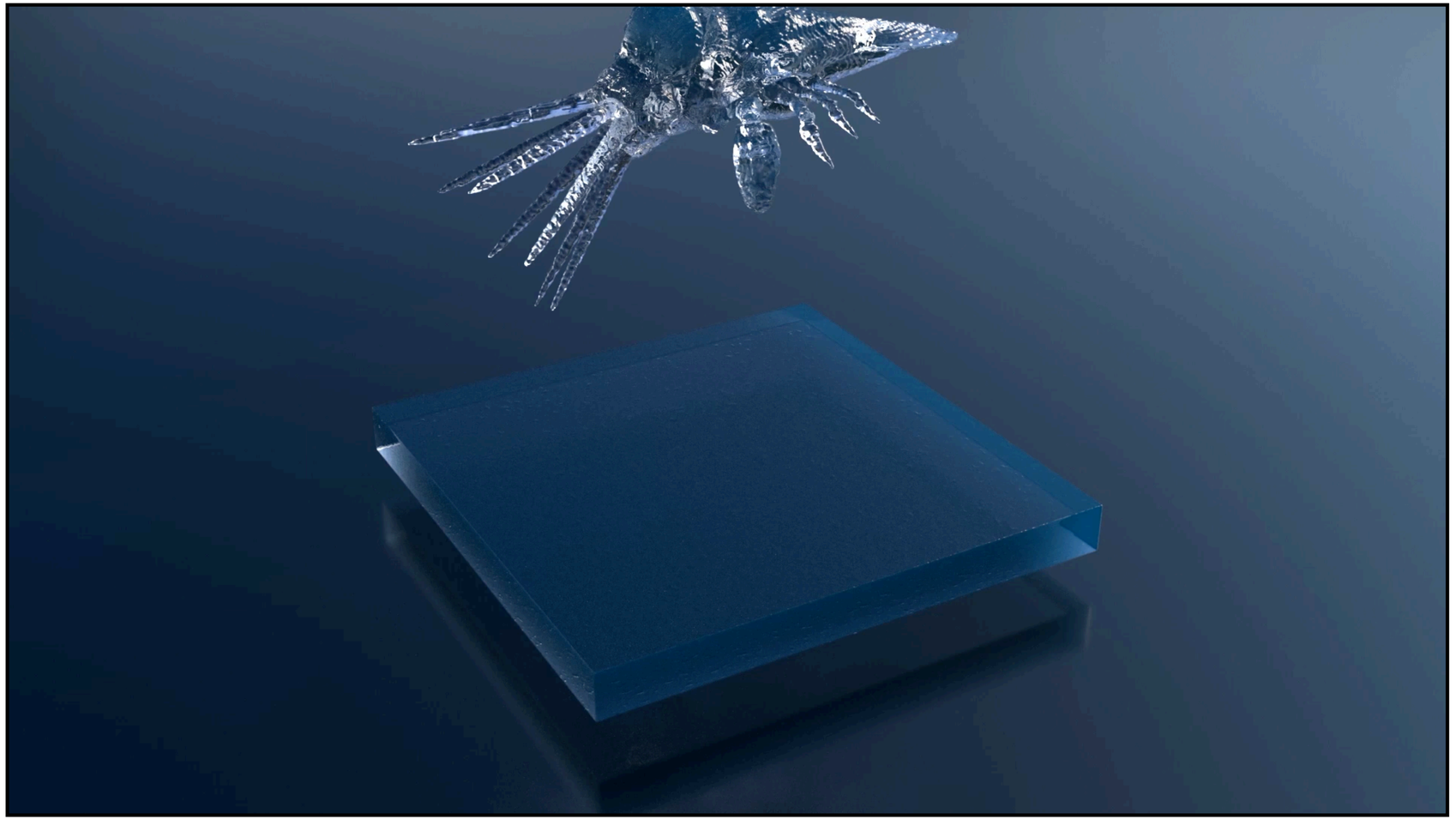
Visual effects ≃ Solving integrals

$$\text{pixel color} = \int \text{light}\, dx$$

Source: DNEG

Numerical Methods for Fluid Simulation

To compute next frame ("timestep"): must solve an enormous linear system



HUGE matrix

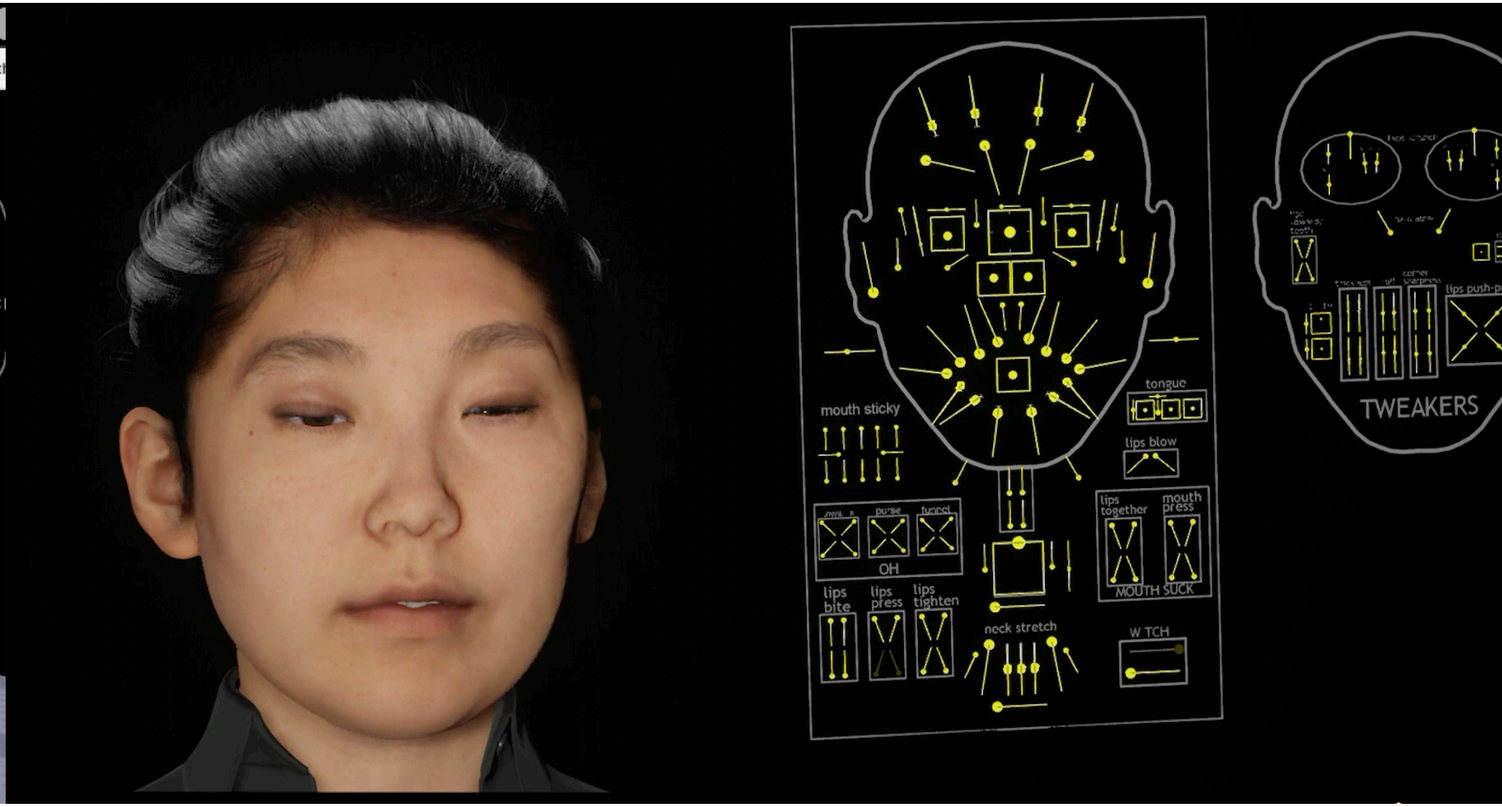
$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

Source: Agora Visual effects

Numerical Methods for Character Animation

Attaching a virtual body to a skeleton using optimization

Source: MetaHuman control rig



Source: Eisko digital doubles

$$\mathbf{x} = \mathbf{A}^{+}\mathbf{b}$$

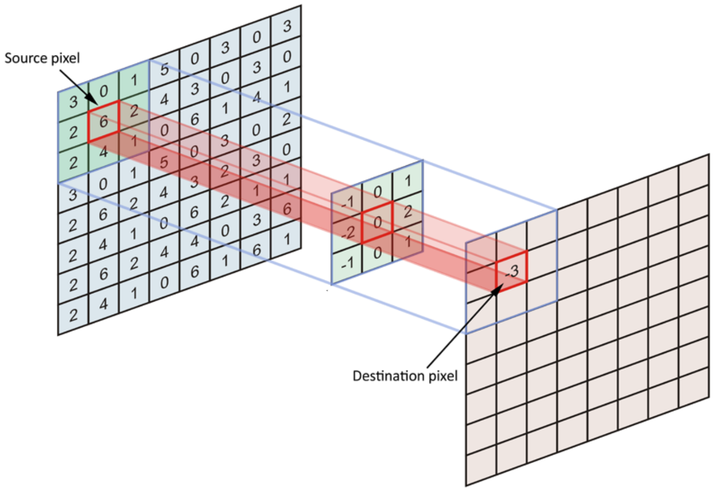
Pseudoinverse

Numerical Methods for Computer Vision

Quickly process images in realtime using learned convolutions ("CNNs")

[Source: Meta, Segment Anything ("SAM")]

Convolution (a particular type of linear transformation)



Numerical Methods for Machine Learning

Computing derivatives of software for fun and profit

[Source: Progressive Growing of GANs (NVIDIA), 2017]

f is a computer program

$$\frac{\partial}{\partial \mathbf{x}} f(\mathbf{x})$$

Hmm..?

A certain feeling of familiarity

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

MATH-111 (Linear Algebra)

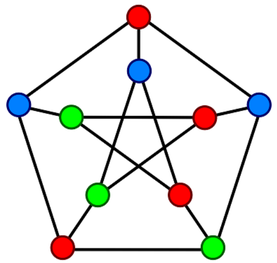
$$\frac{\partial}{\partial \mathbf{x}} f(\mathbf{x}) \quad y = \int_{\Omega} f(x)\,\mathrm{d}x$$

MATH-101 (Analysis I)
MATH-106 (Analysis II)

CS-328 is about the *computer implementation* of these abstract concepts.

Typical Computer Science vs. Numerical Methods

(One way to think about it)

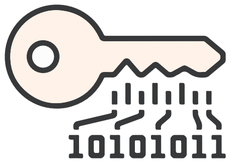


Graph theory

$$\mathbb{N} \longrightarrow \mathbb{R}^n$$

Databases

Network protocols



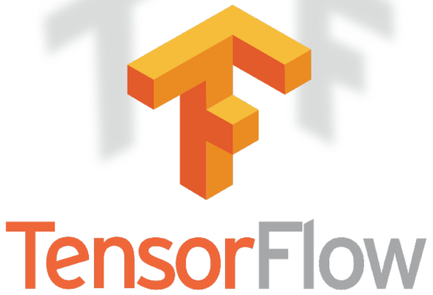
Cryptography

Midjourney: *Isaac Newton watching an apple falling from a tree. Comic style on white background.*

Many layers

- Applications
- Frameworks
- Optimization algorithms
- Matrix factorizations
- Compilation
- Floating point arithmetic
- Processor architecture

lower level



Many layers

- Applications
- Frameworks
- Optimization algorithms
- Matrix factorizations
- Compilation
- Floating point arithmetic
- Processor architecture

lower level

Gradient descent

Newton's method in N dimensions

Broyden's Method

Gauss-Newton method

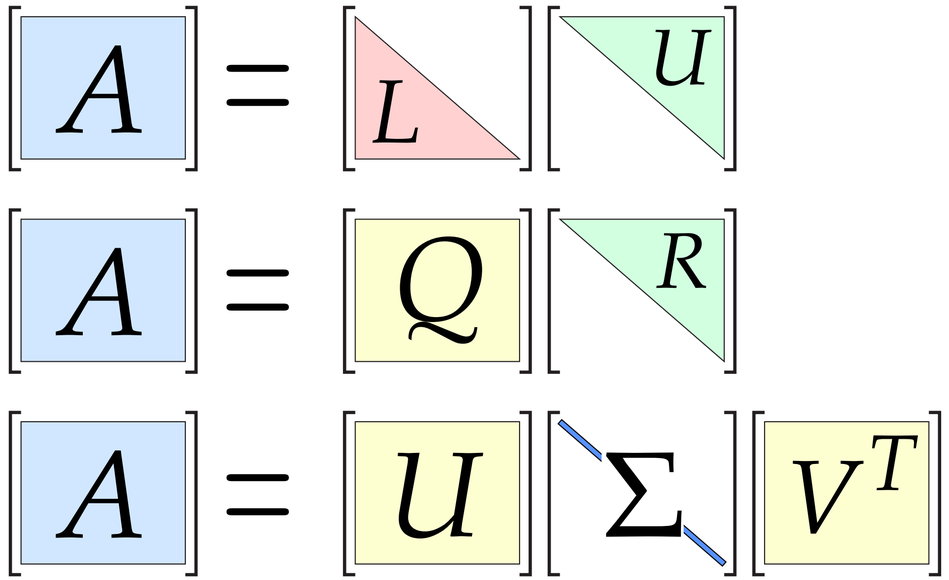
Levenberg-Marquardt

Many layers

- Applications
- Frameworks
- Optimization algorithms
- Matrix factorizations
- Compilation
- Floating point arithmetic
- Processor architecture

lower level

"Big 3" factorizations: LU, QR, SVD



Many layers

- Applications
- Frameworks
- Optimization algorithms
- Matrix factorizations
- Compilation
- Floating point arithmetic
- Processor architecture

lower level

Compilation of floating point math

Just-in-time compilation (JAX, XLA, etc.)

Many layers

- Applications
- Frameworks
- Optimization algorithms
- Matrix factorizations
- Compilation
- Floating point arithmetic
- Processor architecture

lower level

Today's topic

History

IEEE-754 arithmetic

Error propagation

Many layers

- Applications
- Frameworks
- Optimization algorithms
- Matrix factorizations
- Compilation
- Floating point arithmetic
- Processor architecture

lower level

Superscalar processors

Vectorization

Memory hierarchy

Shared memory (GPUs)

(and how this all relates to numerical methods)

Python

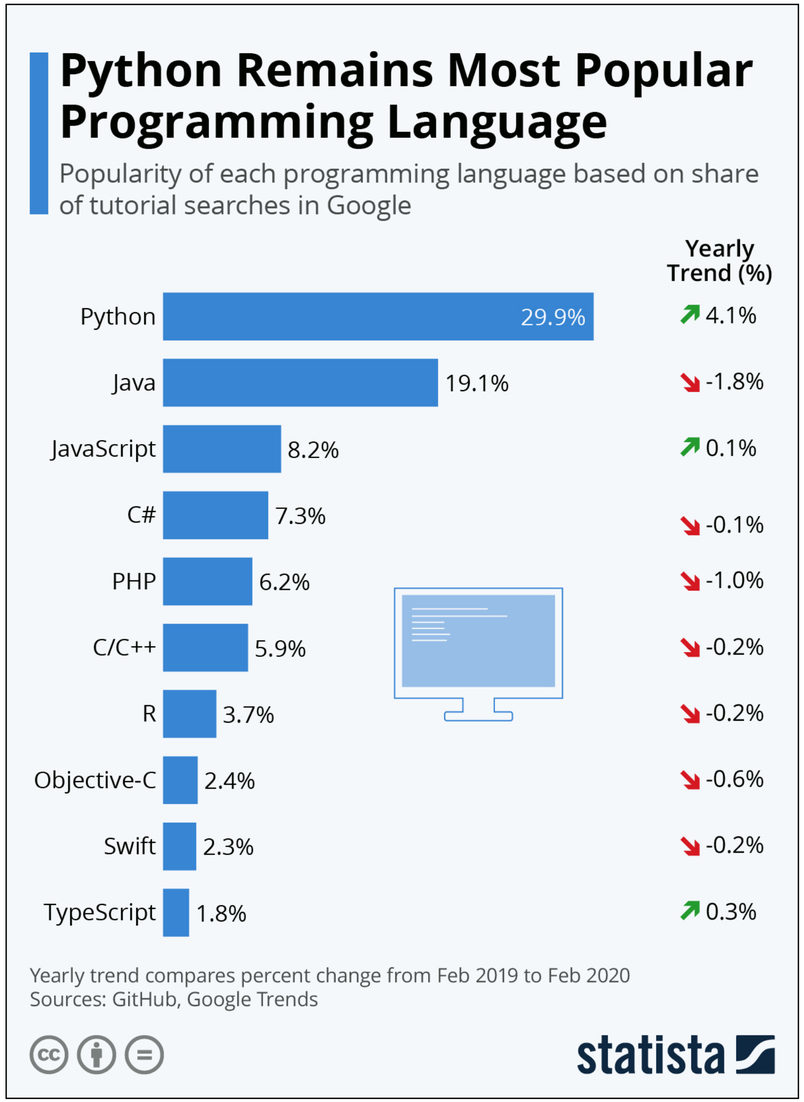
- Python isn't a perfect programming language.
- But it is pretty good.

  → Many people use it.

  → You should probably know how to use it.

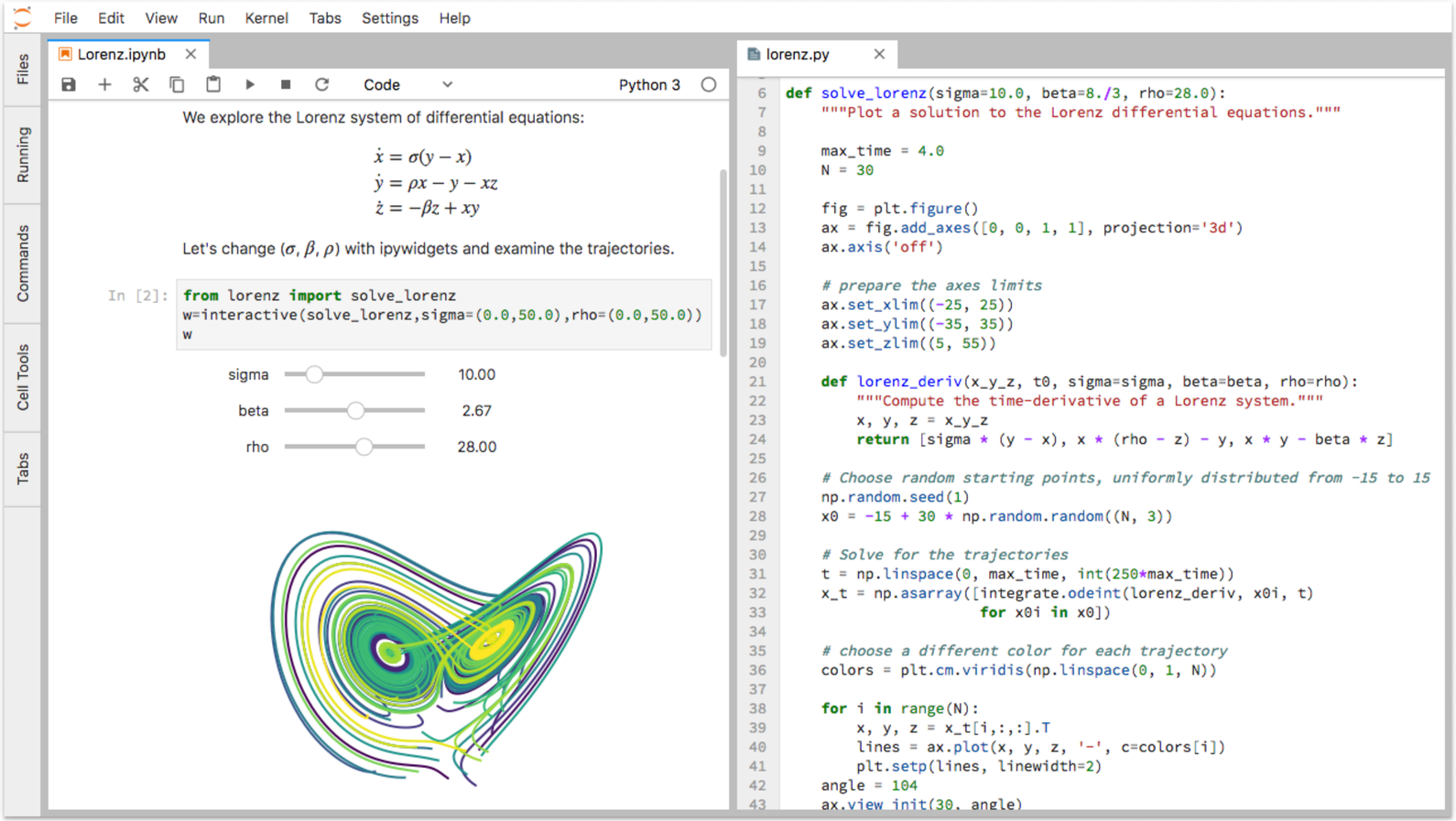
  *You **will** know how to use Python after taking CS328.*

- Our Python reference version: **3.13**
  (Standard Python or Conda, does not matter)
- Python is an interpreted language with an unusual identation-based syntax.



Source: Statista

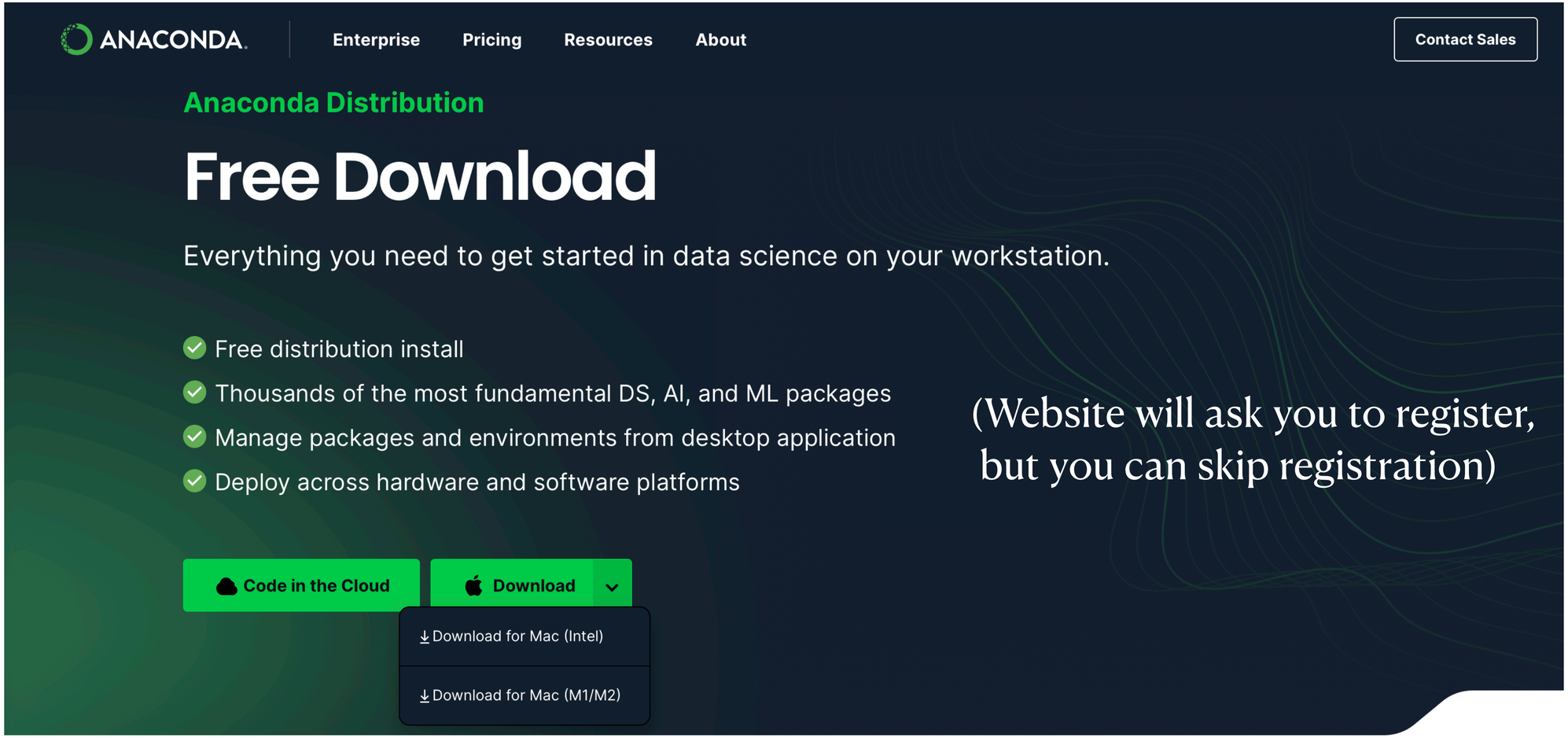
Jupyter lab



When TAs and I talk about "Jupyter notebook(s)", we mean: *Jupyter Lab*. There is also an old & deprecated Jupyter notebook tool, please don't use that.

Setting up Python

Reasonable choice if you don't already have Python on your machine



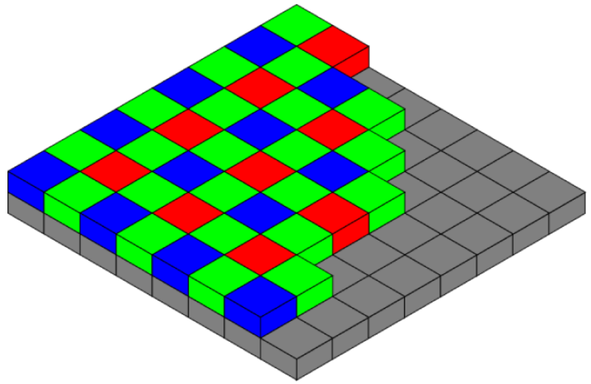
(Website will ask you to register, but you can skip registration)

Link: https://www.anaconda.com/products/individual

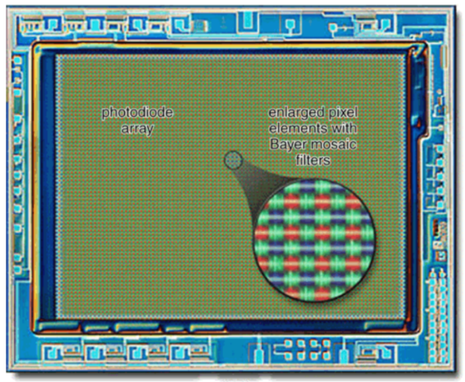
Homework 1: Basics

- Getting started with NumPy, SciPy, Matplotlib
- Floating point arithmetic
- Error Analysis
- First steps with a Monte Carlo method

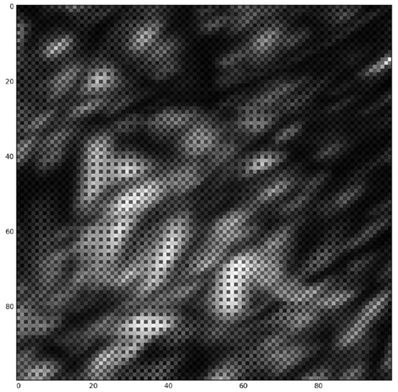
Homework 2: processing a RAW image

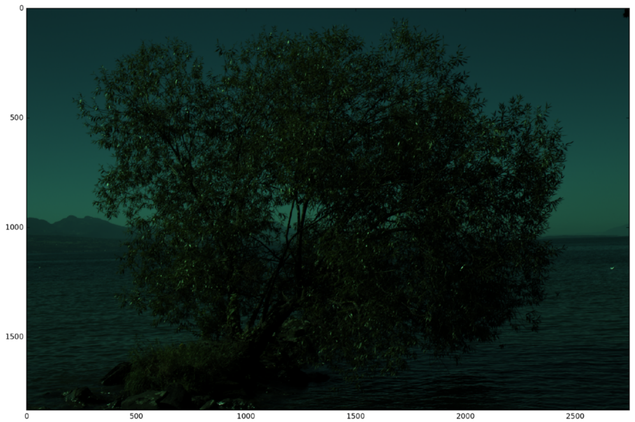


The Bayer grid: Interleaved placement of R/G/B pixels on the sensors of current digital cameras [Wikipedia]



Microscope image of an actual Bayer grid on a CCD sensor [Kenneth R. Spring]

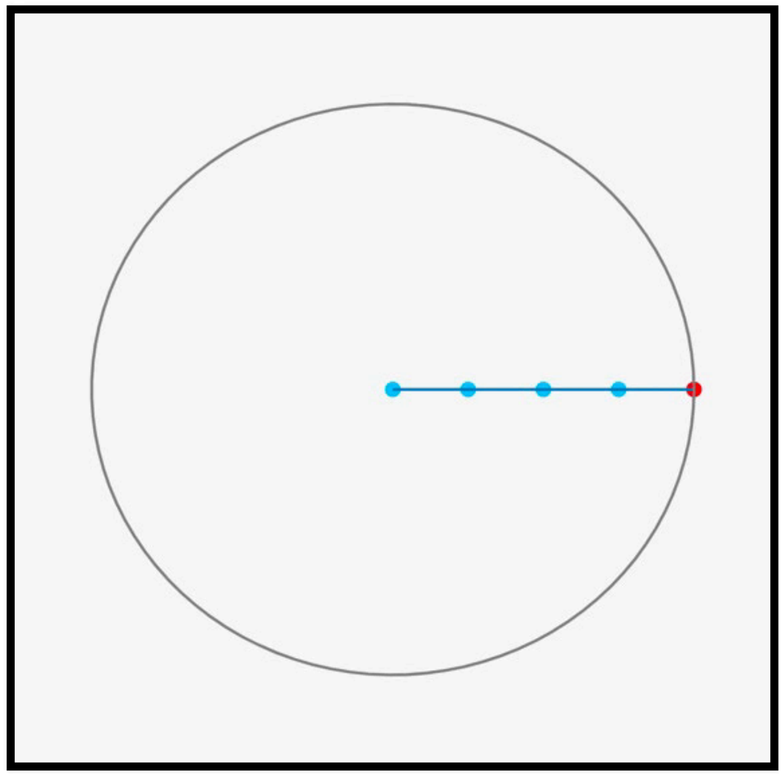




Task: Demosaic & process a RAW Bayer grid data from a Canon DSLR camera.

Solve a least squares problem to go from "Native Camera" color space to Human (sRGB) color space.

Homework 3: Inverse Kinematics



Simple IK "chain" to simulate a worm body being pulled

Singular Value Decomposition (SVD) to solve a rank-deficient linear system

Homework 4: Autodiff & machine learning

- We will build a tiny machine learning framework and use it to train a tiny neural network.

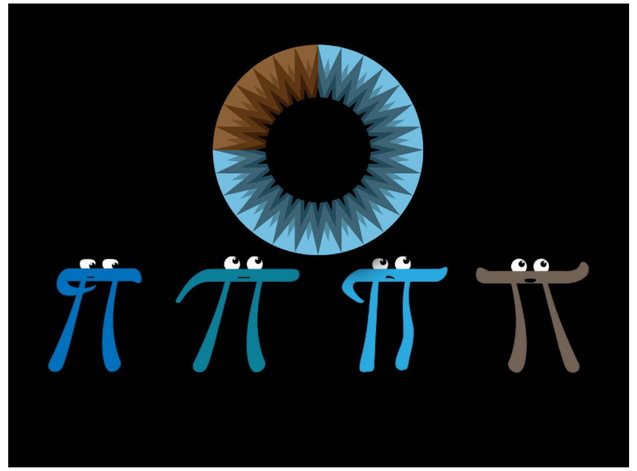
- The goal is to understand what happens inside big frameworks like PyTorch, TensorFlow, etc., that everyone is using these days.

⚠️ A few words of caution ⚠️

- This is a *survey course*. In other words: it covers many different methods, but **does not go deep** into any particular topic.
- We will give a bit of guidance to get started with Python, but the rest is *your job*. (This is is *not* a programming language course.)
- BA3 students find this course *relatively hard* compared to BA5 students.
- If you aren't interested in numerical methods, then you will not like this course.

Additional resources

- *Essence of Linear Algebra* (Grant Sanderson aka. YouTube user 3Blue1Brown)



- **No text book** (will distribute readings for those interested in learning more about a topic. Those readings are **optional**.)

Rules and procedures: communication

- The exercise session is the **main place** to ask questions about the content & homework.
- You can also reach the team asynchronously via the **EdStem** link on Moodle.
  - "Best-effort": please be kind and don't expect responses, e.g., during weekends/evenings/vacations.
  - Everything is public. You can also chime in!
  - Do not post your homework solutions here.
- For administrative questions: contact me directly.
- We will use Moodle for all homework submissions and announcements. Ensure that you are subscribed and receive announcement messages.
- Lecture slides on Moodle: **2 versions** (one with animations, one without).
- Lecture recordings: posted **with 1-2 day delay**.

Rules and procedures: academic integrity

- You are welcome to chat with your friends about CS328 material. But you may not show your own solution code, or look at / copy the solutions of others.
- A large language model counts as a friend 🤖. You may not use an LLM to do your homework.

(ChatGPT/Claude/Cursor/.. are tempting, because they can easily solve bachelor-level CS problems. If you use them for this, you will become dependent on access to LLMs. Remember that the final exam tests what YOU can do with YOUR brain 🧠 alone.)

Rules and procedures: late policy

- Homeworks must be submitted by the cutoff time posted on Moodle.
  Please don't upload seconds before the deadline to avoid accidents.
- Double-check that you indeed uploaded the correct file.
- 25% penalty for each extra started late day (i.e. no points after 4 days).
  Here, *late* := as defined by Moodle
- Flexible in the case of emergencies
  (please reach out before the deadline in such cases.)

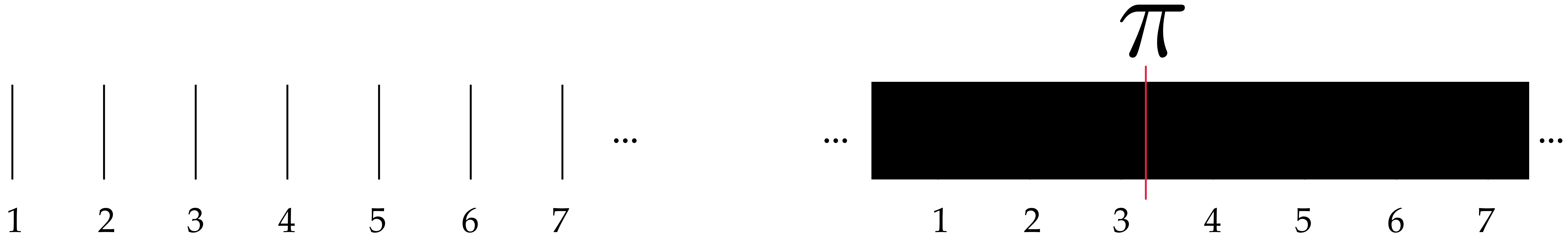
Rules and procedures: grading policy

- Continuous control during the semester with 4 homeworks. **35% of final grade.**
- Pen &paper exam during January exam session. **65% of the final grade.**
- **Visiting students**: remotely proctored exams are possible if your home university has the infrastructure for this, but it's your job to set it up.
- **Fine print:** homework and exam are graded on a 1-6 scale and then averaged with these weights. The mapping from raw percentages to the 1-6 scale generally uses a **curve** (i.e.: an affine transformation based on the distribution of grades).

IEEE 754 Floating Point Arithmetic

(What a strange name?)

$$\mathbb{N} \longrightarrow \mathbb{R}^n$$



3.141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825 3
4211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964 46
2294895493038196442881097566593344612847564823378678316527120190914564856692346034861045 43266
4821339360726024914127372458700660631558817488152092096282925409171536436789259036001133053 05
488204665213841469519415116094330572703657595919530921861173819326117931051185480744623799627
495673518857527248912279381830119491298336733624406566430860213949463952247371907021798609437
027705392171762931767523846748184676694051320005681271452635608277857713427577896091736371787
214684409012249534301465495853710507922796892589235420199561121290219608640344181598136297747

An infinite amount of storage for *one number*..

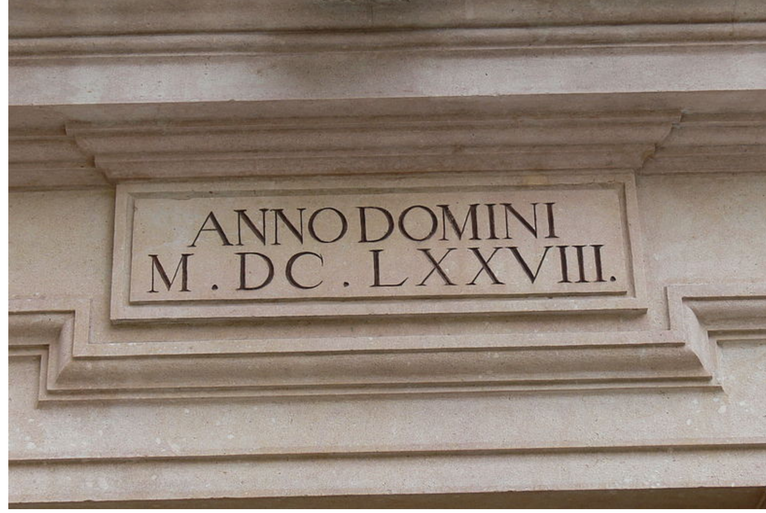
- The number π is *sort of important*. But we will not be able to store it.
- There are infinitely many numbers of this type between any two numbers.
- Some sort of compromise will have to be made.

- **The compromise**: the program will no longer compute the correct answer.

(Wait, what ?!)

Number Systems — Integers

Two flavors you have encountered so far



Roman numerals (Wiki commons)

Positive integers:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	0	0	0	1	0	1

$= 2^7 + 2^6 + 2^2 + 2^0 = 197$

Signed integers (two's complement)

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	0	0	0	1	0	1

$= -2^7 + 2^6 + 2^2 + 2^0 = -59$

Number Systems — Integers

Typical sizes

`uint8`	0	255
`int8`	-128	127
`uint16`	0	65535
`int16`	-32768	32767
`uint32`	0	4294967295
`int32`	-2147483648	2147483647
`uint64`	0	18446744073709551615
`int64`	-9223372036854775808	9223372036854775807

Number Systems — Fixed point arithmetic

Basic idea: allow negative exponents

2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
1	1	0	0	0	1	0	1

$= 2^3 + 2^2 + 2^{-2} + 2^{-4} = 12.3125$

Drawback 1: big numbers require a lot of storage. For example,
$4 \times 10^{80} = 400000000000000000000000000000000000000000000000000000000000000000000000000000$ (~ 268 bits)

Drawback 2: growth of exponent range during operations

$$\left(\sum_{i=-k}^{k} a_k 2^k \right) \left(\sum_{i=-k}^{k} b_k 2^k \right) = a_{-k} b_{-k} 2^{-2k} + \cdots + a_k b_k 2^{2k}$$

Number Systems — Rational numbers

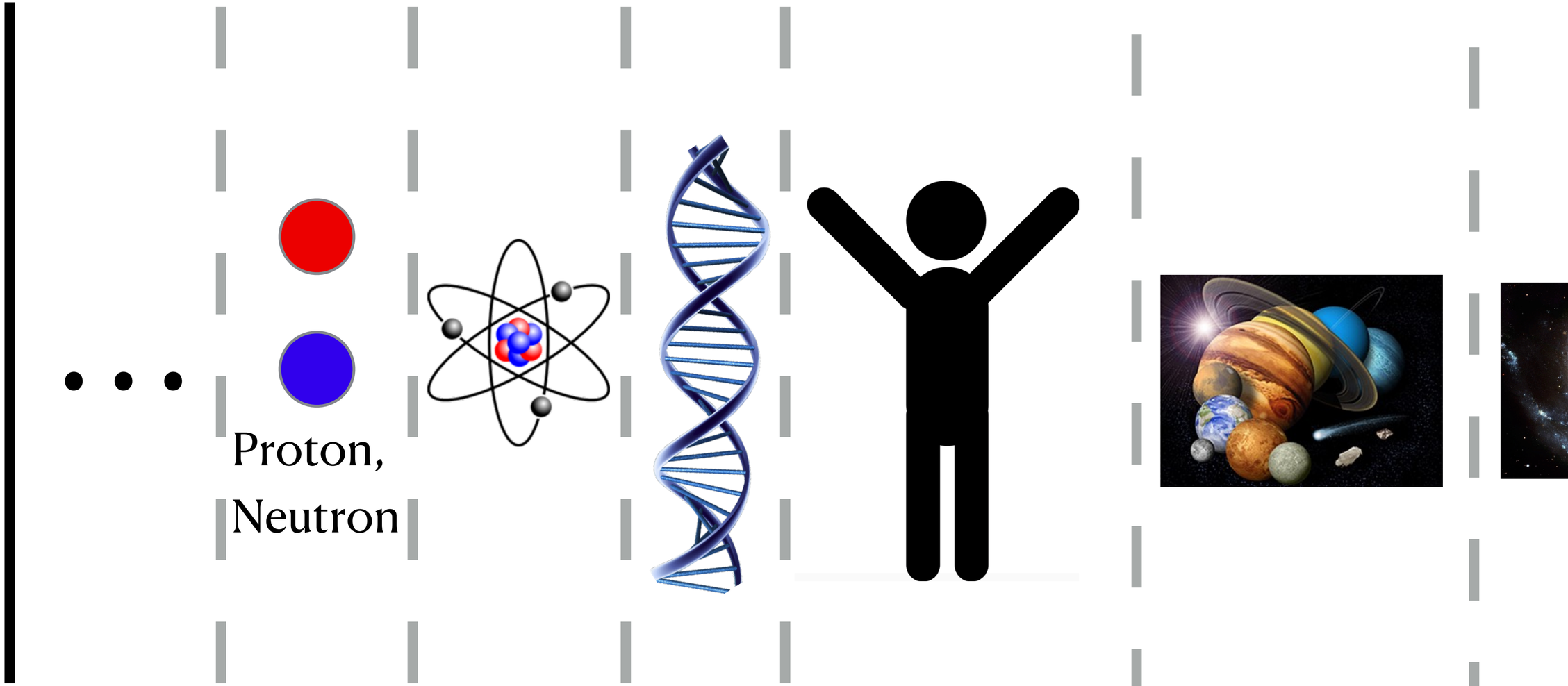
$$\mathbb{Q} = \{a/b : a, b \in \mathbb{Z}\}$$

Exact representation:

$$a/b + c/d = (ad+cb)/bd$$

As we discussed, this is generally not a good thing.

Different scales

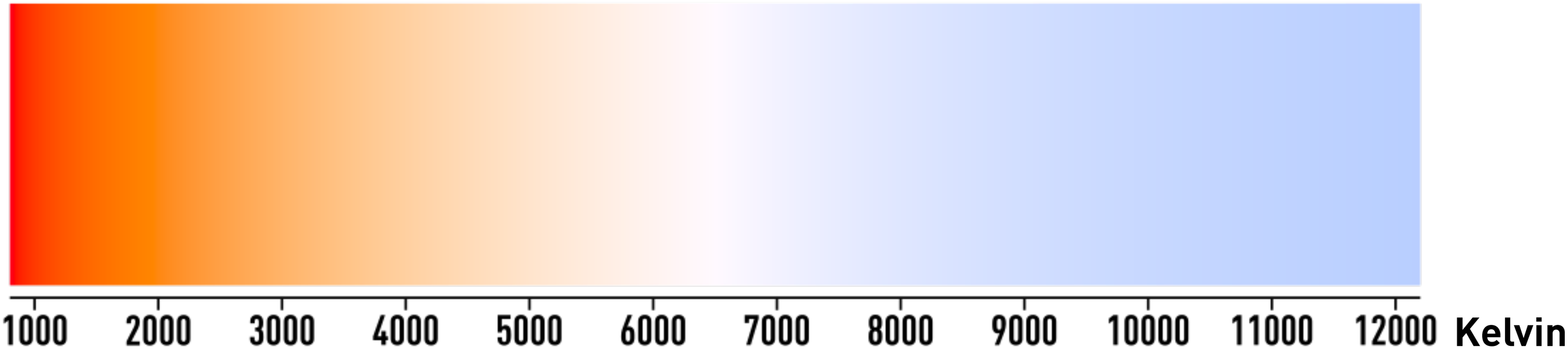


10^{-33} cm

Planck length

Thermal radiation: Planck's black body law

[Wiki commons]



$$B_\nu(T) = \frac{2h\nu^3}{c^2} \frac{1}{e^{\frac{h\nu}{kT}} - 1}$$

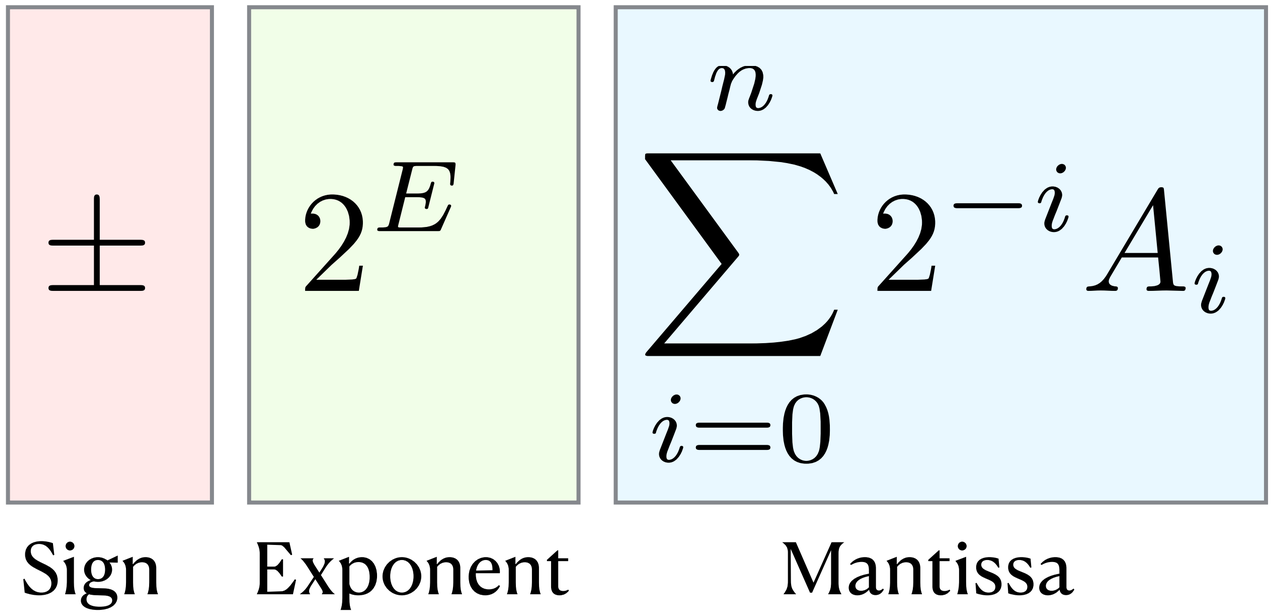
h: $6.626070040 \times 10^{-34}$

c: 299 792 458

k: $1.38064852 \times 10^{-23}$

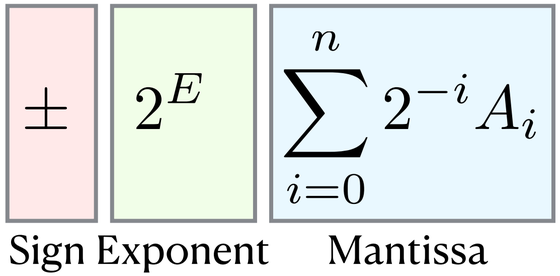
Number Systems — Floating point numbers

"Baby version"



Number Systems — Floating point numbers

"Baby version"



$$1.0 = +\ 2^{0}\ (2^{-0}) = +\ 2^{0}\ 1.00000_2$$

$$0.5 = +\ 2^{-1}\ (2^{-0}) = +\ 2^{-1}\ 1.00000_2$$

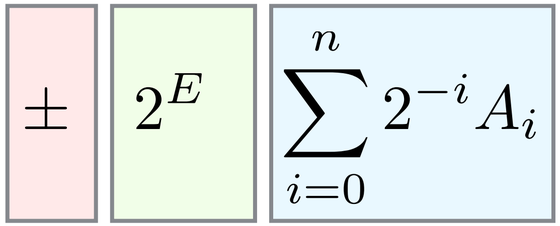
$$-0.75 = -\ 2^{-1}\ (2^{-0}+2^{-1}) = -\ 2^{-1}\ 1.10000_2$$

$$3.14 \approx +\ 2^{1}\ (2^{-0}+2^{-1}+2^{-4}+2^{-7}) = +\ 2^{1}\ 1.10010010_2$$

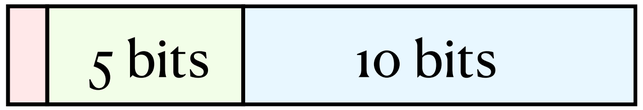
This digit is always the same?

Number Systems — Floating point numbers

Standard floating point formats



`float16`: ~3 decimal digits. Range: -65504 .. 65504



🎮 "half precision"

`bfloat16`: ~2 decimal degits. Range: - 3.4×10^{38} .. 3.4×10^{38}

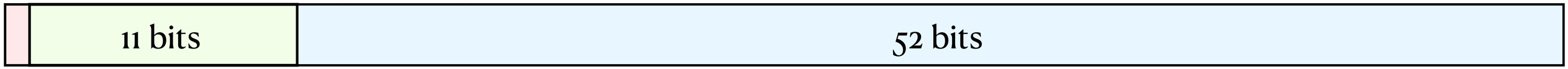
🧠 "brain float"

`float32`: ~7 decimal digits. Range: - 3.4×10^{38} .. 3.4×10^{38}

🎮 "single precision"

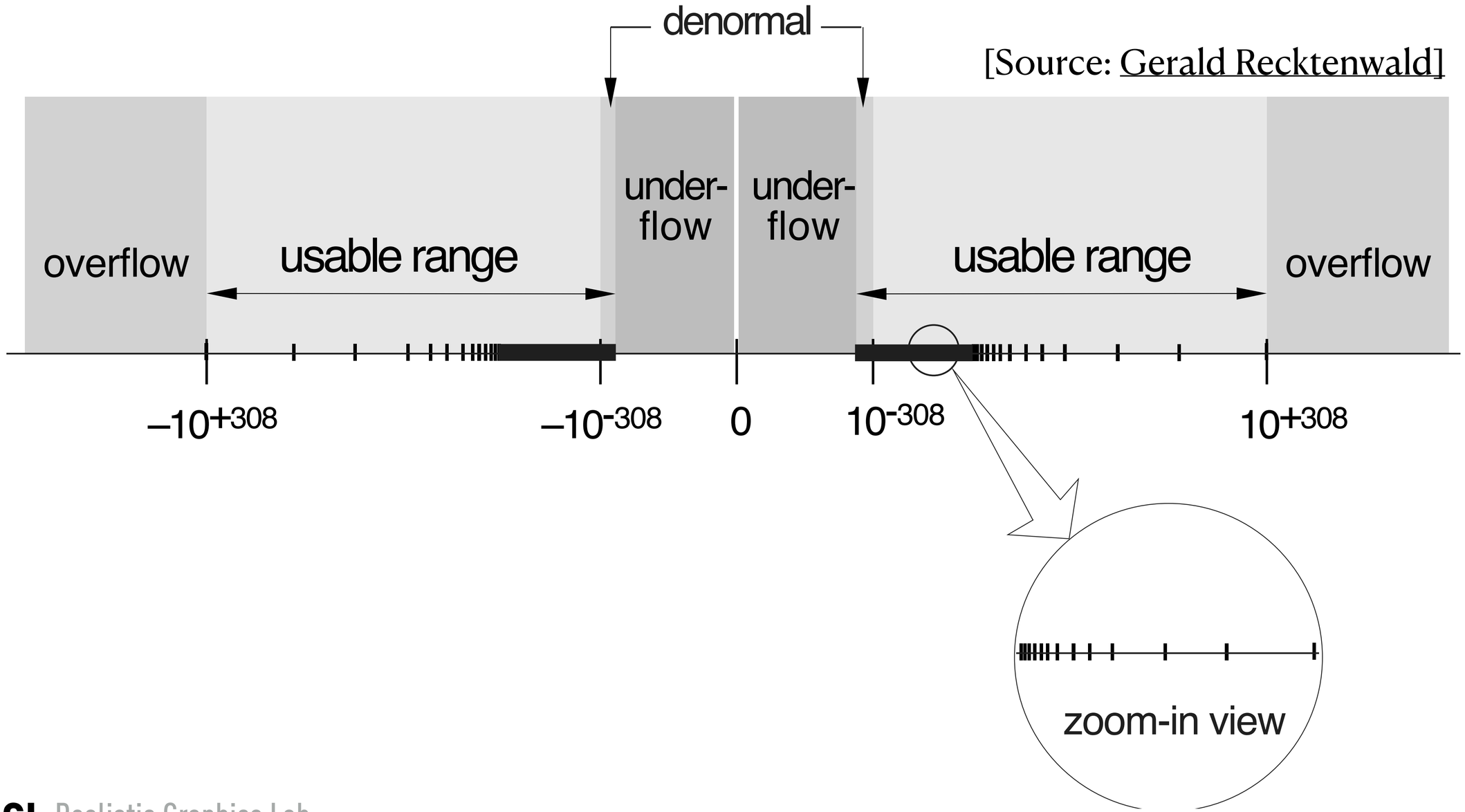
`float64`: ~16 decimal digits. Range: - 2.22×10^{308} .. 2.22×10^{308}

"double precision" (serious applications!) 👩‍🔬



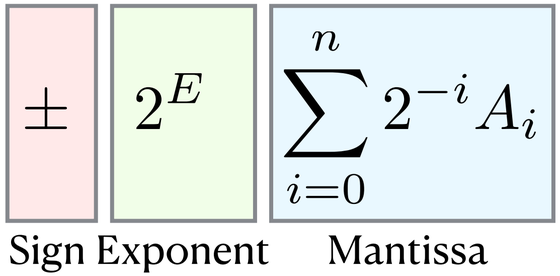
(Please don't memorize these numbers for the exam.)

Number resolution changes with magnitude



Number Systems — Floating point numbers

Problems with the baby version



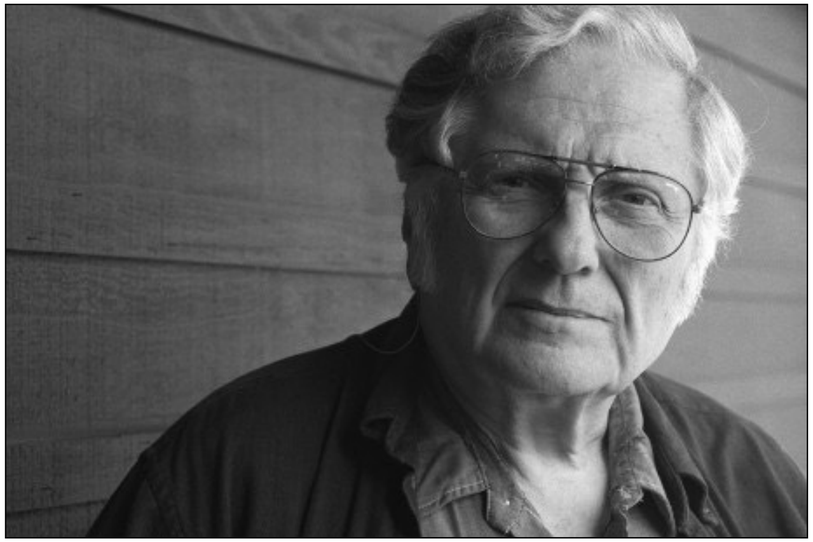
- How to represent zero?
- What if the number becomes too big during a calculation?
- What if the number becomes too small during a calculation?
- What if the user is trying to do a nonsensical calculation?
- What to do when the result of a calculation cannot be represented exactly?

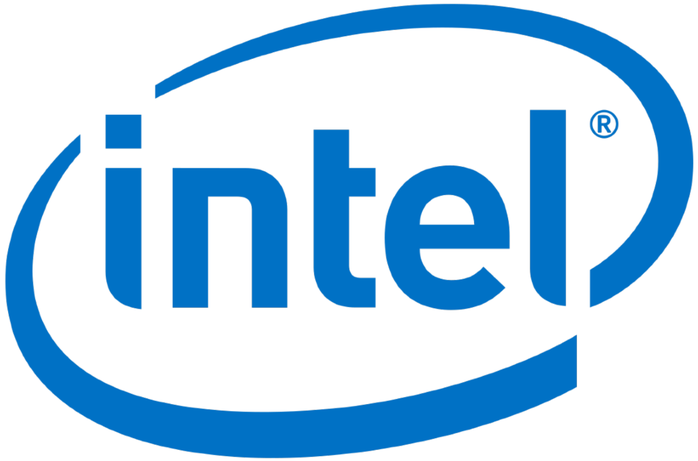
The history of floating point computing machinery

Let's go computer shopping, 70s style..

The IEEE 754 Standard

Enter William Kahan





- Published in 1985 (but developed for many years)
- Super-ambitious proposal at the time
- Behavior is **fully specified**: every arithmetic operation produces a result, whether it is mathematically expected or not, and exceptional operations raise a signal

Demo time

IEEE 754: Nitty Gritty details

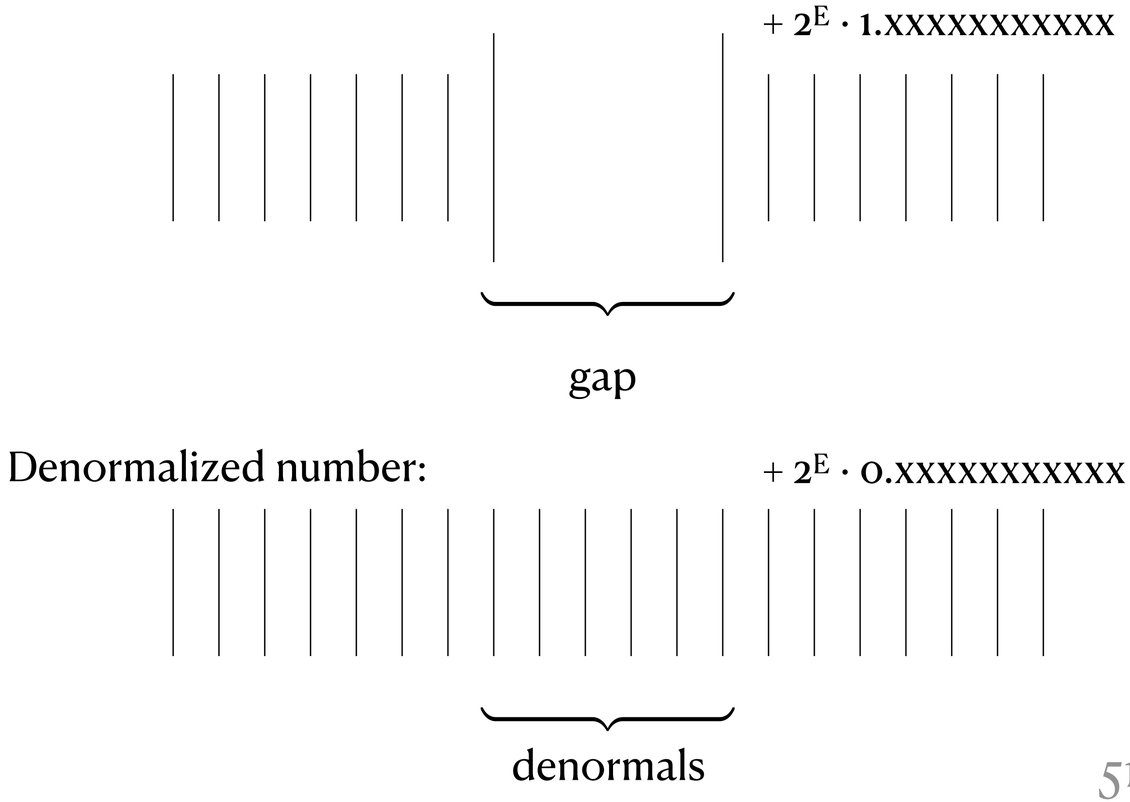
- How to represent zero?
  - *Easy, just don't store the leading one. Zero is signed in IEEE754* 😱.
- What if the number becomes too big during a calculation?
  - *Return special number "infinity" (can be positive/negative)*
- What if the number becomes too small during a calculation?
  - *Gradual underflow through denormalized numbers.*
- What if the user is trying to do a nonsensical calculation?
  - *Return special number "NaN" (Not a Number). NaN is infectious* 🦠
- What to do when the result of a calculation cannot be represented exactly?
  - *Use Banker's rounding (round-to-nearest-tie-to-even) by default, other modes available.*

IEEE 754: Nitty Gritty details, contd.

Major innovations of this standard compared to previous ones

- **Accuracy guarantees**
  - *Elementary arithmetic operations (+, -, *, /, sqrt)* done in infinite precision and then *rounded* to a representable number.

- **Denormalized numbers**
  - Enables gradual underflow.
    Most controversial feature of IEEE 754
  - Handled badly by current processors,
    can make a calculation 100-1000x slower.



IEEE 754: Nitty Gritty details, contd.

Special case rules

*	-Inf	-o	o	Inf	NaN
-Inf	Inf	NaN	NaN	-Inf	NaN
-o	NaN	o	-o	NaN	NaN
o	NaN	-o	o	NaN	NaN
Inf	-Inf	NaN	NaN	Inf	NaN
NaN	NaN	NaN	NaN	NaN	NaN

Common sense, e.g.:

- (-Inf) * (-Inf) = Inf
- Inf * x = sign(x) * Inf
- Inf * o = NaN
- NaN * x = NaN

Other subtleties: comparison against NaN is always `false`

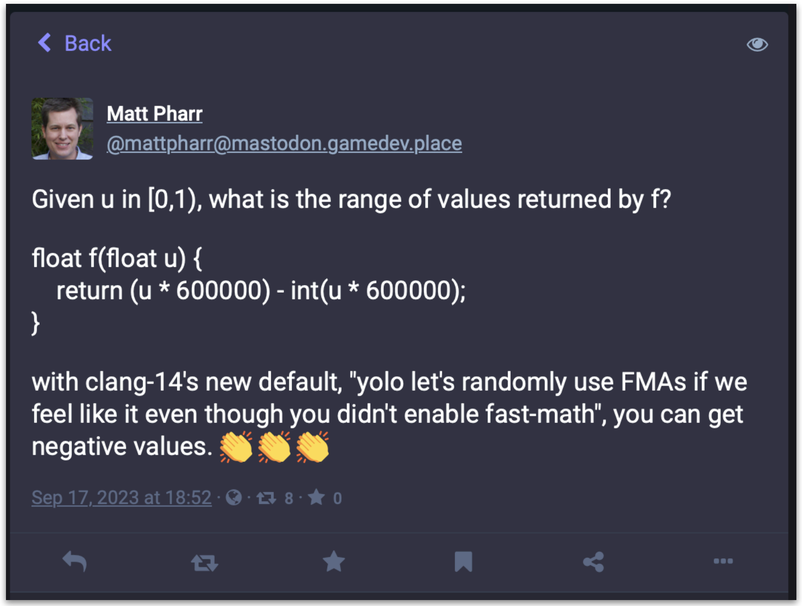
what's the difference?

```
def f(x):
    if x < 0 or x > 1:
        raise Exception("Invalid argument")
    return …
```

```
def f(x):
    if not (x >= 0 and x <= 1):
        raise Exception("Invalid argument")
    return …
```

IEEE 754: Danger zone

- **Transcendental operations** (`sin`, `cos`, `atan`, `exp`, `log`) are much slower (50-300 clock cycles) and less accurate.
- **Fused Multiply-Add** (*FMA*). Computes `a*b+c` directly (faster, with only one rounding step). But now there are two ways to do the same thing.



- **Compiler optimizations** may violate expected IEEE 754 rounding behavior.
- **Denormalized numbers** are often very slow and cause surprises. Can turn them off.
- Careful when converting floats to integers, and vice versa.
- Arithmetic transformations that are "pointless" on pencil and paper can make huge accuracy difference.

Midjourney: *A server in a datacenter exploding.*

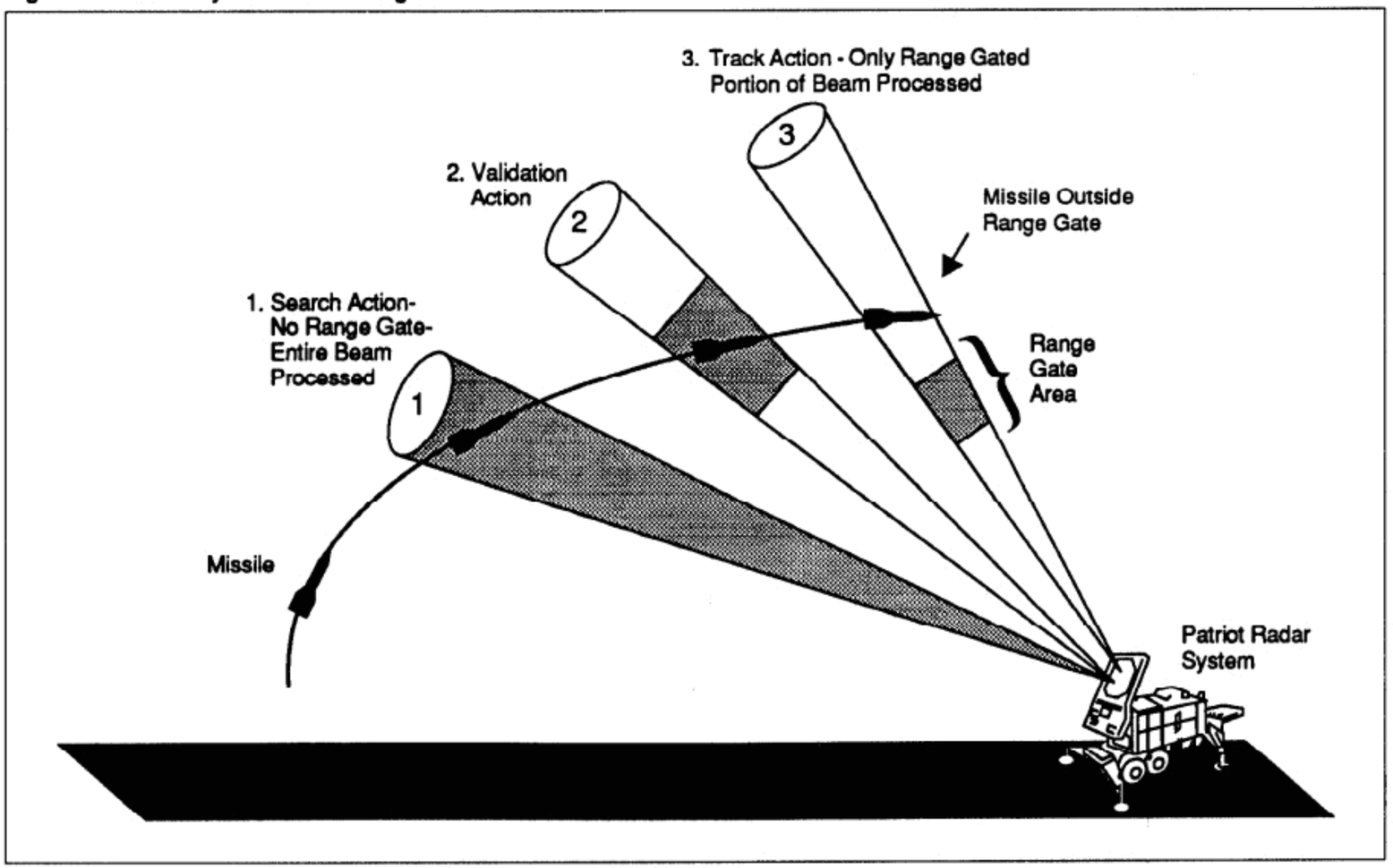
Let's talk about errors

Catastrophic numerical errors

Ariane 5 Rocket failure
June 14, 1996

Patriot missile failure

Figure 5: Incorrectly Calculated Range Gate



Rounding error accumulation

1991: Patriot battery failed to intercept missile in Dharan, Saudi Arabia

28 people died, 96 injured

Model error

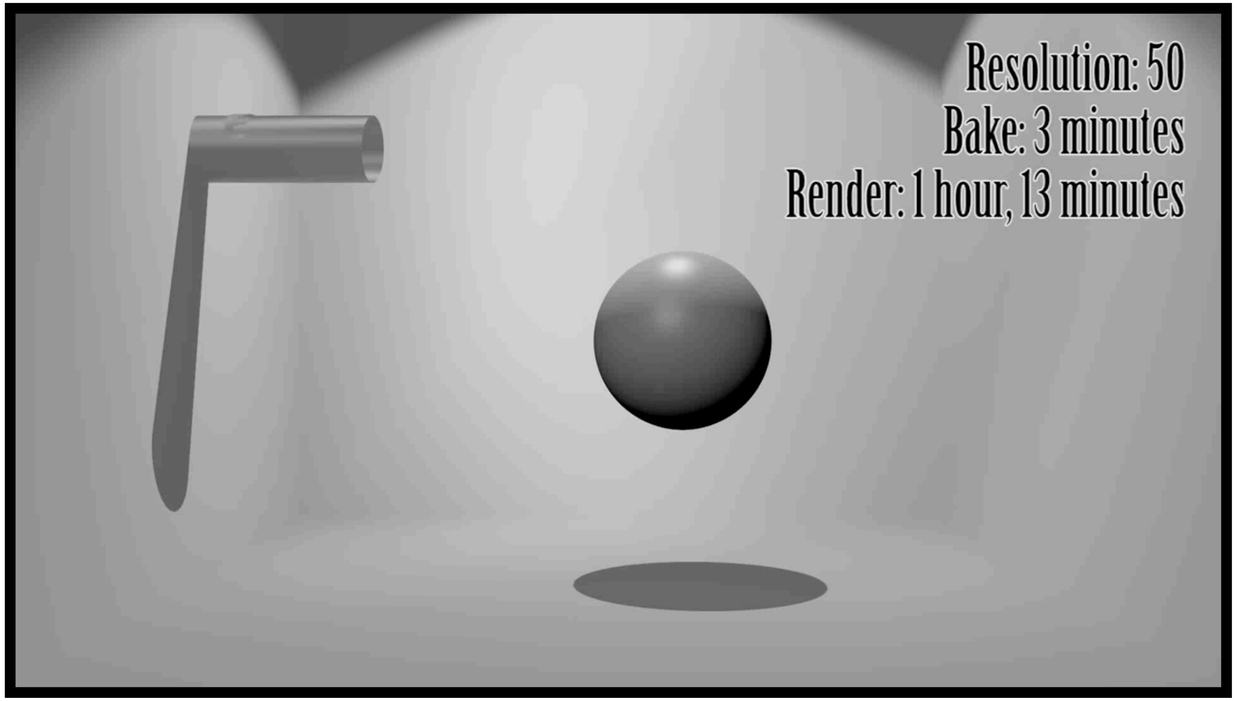
Let's compute the surface of the earth

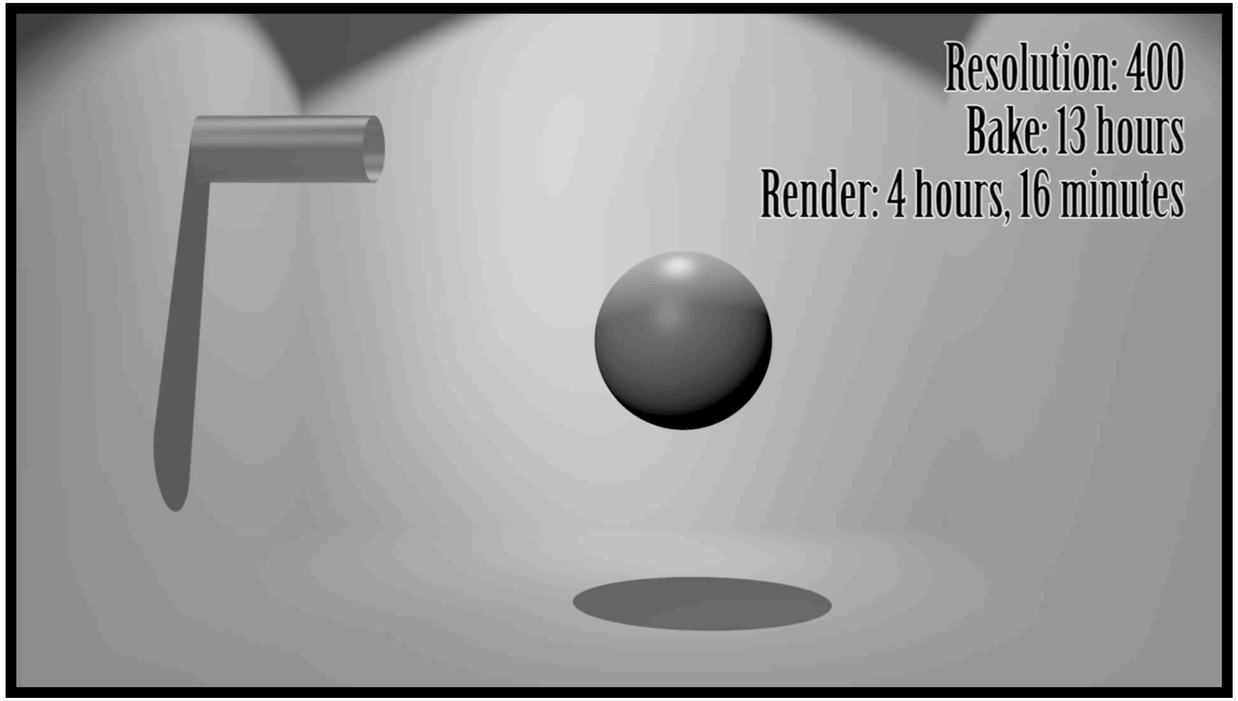
Midjourney: *an egg-shaped planet earth on white background.*

$$A = 4\pi r^2$$

Discretization error

Fluid simulation represented at different spatial resolutions





[YouTube user `Diesel17th`]

This is effectively a type of *model error*.

Truncation error

$$\sum_{i=1}^{10} \frac{1}{3^i} \approx 0.499991532$$

$$\sum_{i=1}^{\infty} \frac{1}{3^i} = 0.5$$

Happens when we approximate an infinite calculation with something finite.

Rounding error

$$\sum_{k=1}^{\infty} \frac{1}{k} = 15.403683\ ?!?!$$

Series value is *infinite* but converges in IEEE-754 arithmetic!

The issue here

(Let's suppose we use 8 mantissa bits after the decimal point)

$$3.14 \approx +\,2^{1}\,(2^{-0}+2^{-1}+2^{-4}+2^{-7}) = +\,2^{1}\;1.10010010_2$$

$$+\;\;1024 = +\,2^{10}\,(2^{-0}) = +\,2^{10}\;1.00000000_2$$

$$+\,2^{1}\;1.100100100_2$$

Shift to align exponent

$$+\;\begin{boxed}\begin{aligned}&+\,2^{10}\;0.00000001100100100_2\\&+\,2^{10}\;1.00000000_2\end{aligned}\end{boxed}$$

$$=\;\;+\,2^{10}\;1.00000000\color{#d0d0d0}{1100100100}_2$$

Dropped (rounding error)

Catastrophic cancellation

- Of all the mathematical operations, one particular case is *especially dangerous*:

$$a - b \quad (a \approx b)$$

- Also happens for $a+b$ if $a \approx -b$
- Often called *catastrophic cancellation*.
- Can have catastrophic cancellation even if the subtraction is done without error. The problem with this operation is that it can greatly magnify the relative scale of errors present in the operands.

An example of catastrophic cancellation

(Let's suppose we use 8 mantissa bits after the decimal point)

$$3.140 \approx +\ 2^{1}\ (2^{-0}+2^{-1}+2^{-4}+2^{-7}) = +\ 2^{1}\ 1.10010010_2$$

$$-\ 3.148 = +\ 2^{1}\ (2^{-0}+2^{-1}+2^{-4}+2^{-7}+2^{-8}) = +\ 2^{1}\ 1.10010011_2$$

$$-\ \boxed{\begin{array}{l} +\ 2^{1}\ 1.10010010_2 \\ +\ 2^{1}\ 1.10010011_2 \end{array}}$$

$$= -\ 2^{1}\ 0.00000001_2$$

$$= -\ 2^{-7}\ 1.0_2$$

Must adjust exponent to re-establish implicit leading 1

This *isn't a wrong answer* given the approximations involved.
But we now only have "1 bit of information" left from what were much more accurate inputs

Another example of catastrophic cancellation

$$f(x) = \frac{1 - \cos x}{x^2}$$

$$f\left(1.2 \cdot 10^{-5}\right) = \frac{1 - 0.9999999999}{1.44 \cdot 10^{-10}}$$

$$= \frac{0.0000000001}{1.44 \cdot 10^{-10}}$$

(uh oh. this is the only bit of accuracy left.)

$$\approx 0.694..$$

BUT: can show that $f(x) < 1/2 \quad (\text{for } x > 0)$

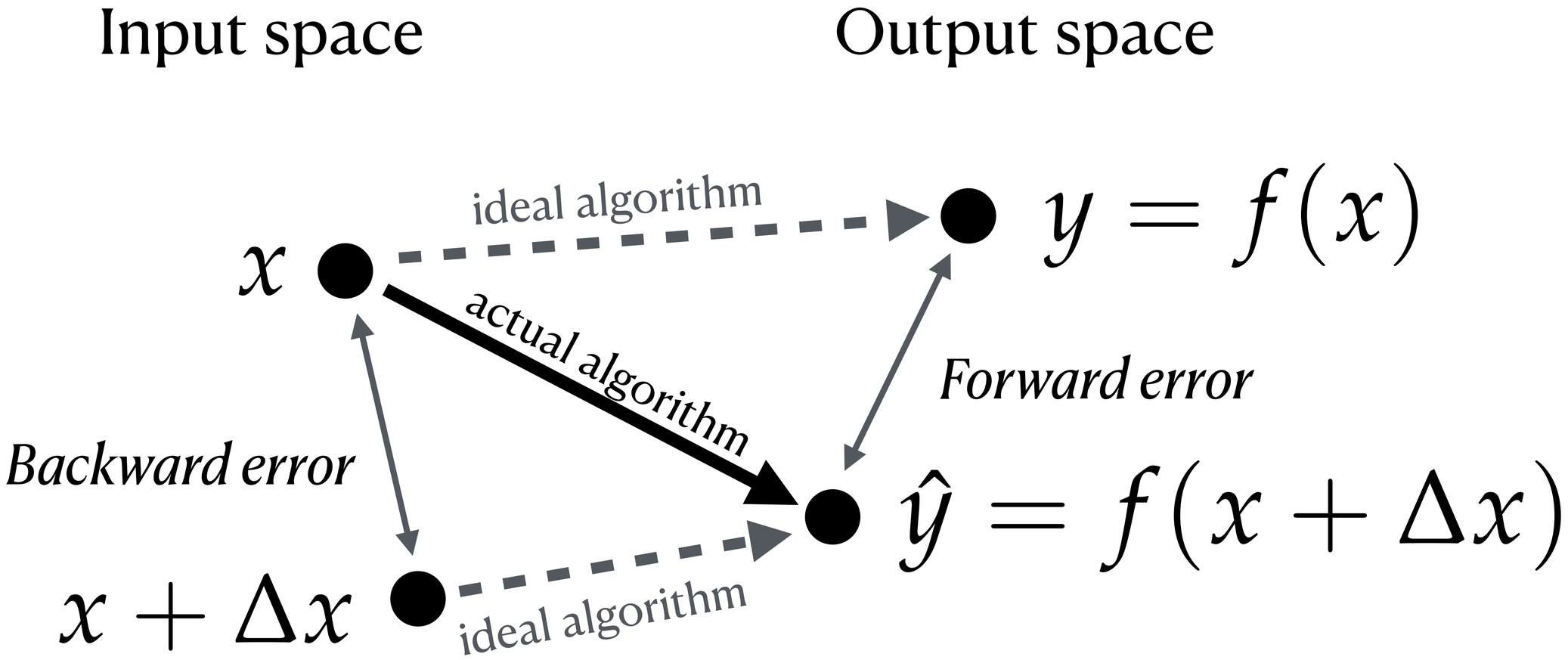
Quantifying error

- **Absolute error**: |true value - approximate value.|
  - Example: 2 cm ± 0.1 cm

- **Relative error**: (absolute error) / |true value|
  - Example: 2 cm ± 0.1 %
  - Another common notation: (true value) (1 ± relative error)

- Error is generally unknown.

Forward and Backward Error



Forward and Backward Error: Example

$$f(x) := \sqrt{x}$$

$$y = f(x)$$

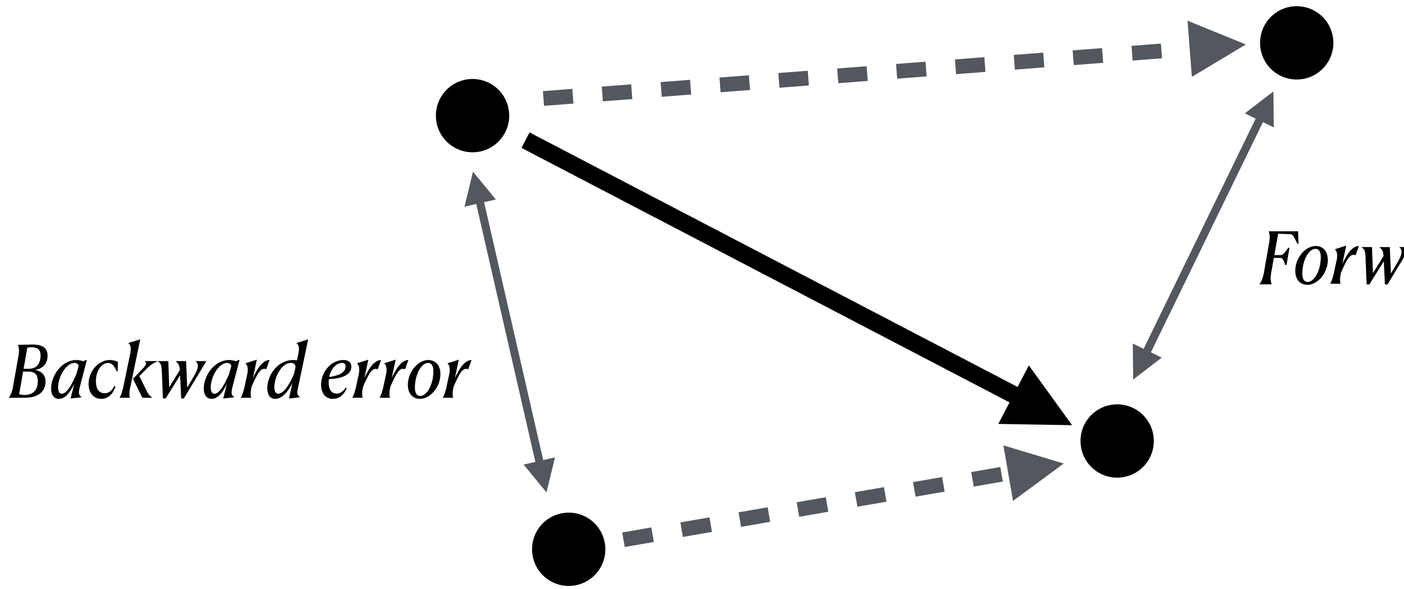
$$\hat{y} = f_{\text{IEEE754}}(x)$$

$$\text{forward error} = y - \hat{y}$$

$$\hat{x} = f^{-1}(\hat{y}) = \hat{y}^2$$

$$\text{backward error} = x - (f_{\text{IEEE754}}(x))^2$$

Conditioning of numerical problems



Condition number: ratio $\dfrac{\text{forward error}}{\text{backward error}}$