Harnessing inexactness in scientific computing

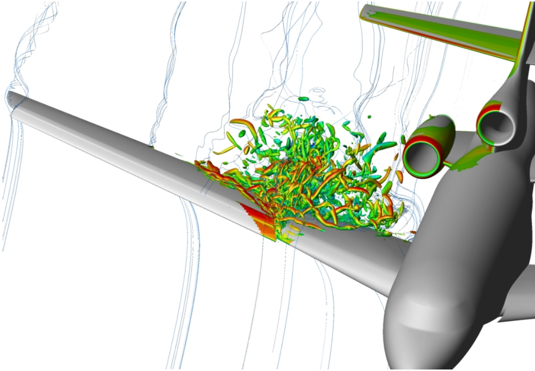
Lecture 1: introduction

Theo Mary (CNRS)
theo.mary@lip6.fr
https://perso.lip6.fr/Theo.Mary/

Elisa Riccietti (ENS Lyon)
elisa.riccietti@ens-lyon.fr
https://perso.ens-lyon.fr/elisa.riccietti/

M2 course at ENS Lyon, 2024–2025

Slides available on course webpage



Unpacking the course

Approximate computing, *vue d'avion*

Ad break

Practical organization

$$\boxed{\begin{array}{c}\textbf{Harnessing inexactness}\\ \textbf{in scientific computing}\end{array}}$$

Harnessing inexactness
in scientific computing

What is scientific computing ?

From traditional science to scientific computing

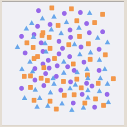
Traditional science

Come up with a theory, then test it experimentally in “real life”

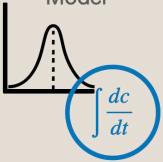
- too difficult
- too costly (planes, cars)
- too slow (climate science)
- too dangerous (defense, drugs)

Scientific computing

Data



Model



Method

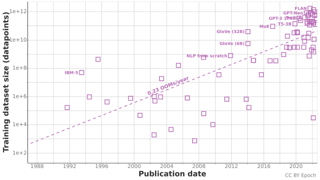
```
i ← 10
if i ≥ 5 then
    i ← i − 1
else
    if i ≤ 3 then
        i ← i + 2
    end if
end if
```

Software

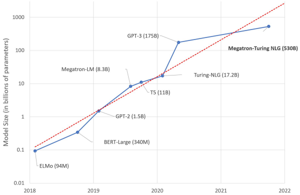
Supercomputer

Current tendency: scale everything up!

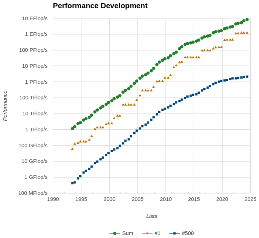
Increasingly large datasets



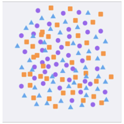
Increasingly complex models



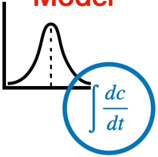
Increasingly powerful computers



Data

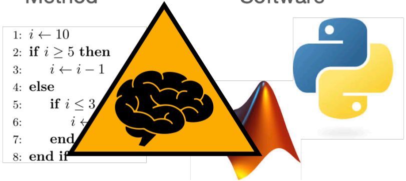


Model



Method

Software



Supercomputer

Let's unpack the title (2/3)

Harnessing inexactness
in scientific computing

Harnessing inexactness
in scientific computing

What is inexactness ?

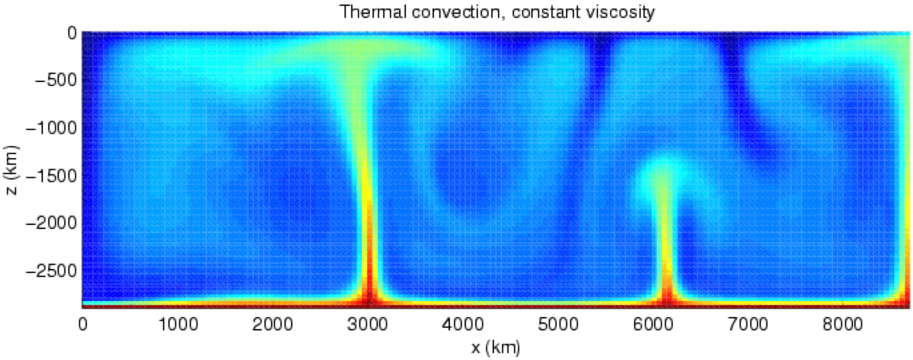
Sources of error in computing

1. **Model errors**
2. **Errors for model deployment**
3. **Rounding errors**

Models guided by the physical knowledge

Physical phenomenon $\longleftrightarrow$ Mathematical equation

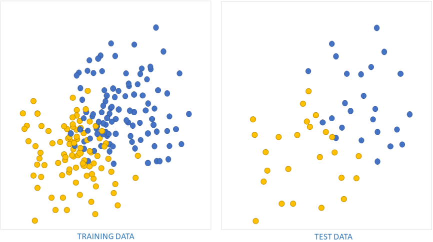
$$\text{Heat propagation}: \frac{\partial u}{\partial t} = \Delta u$$

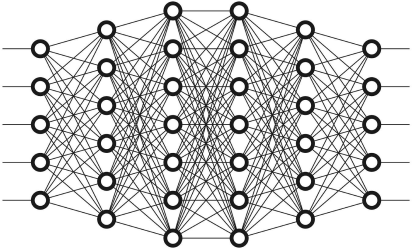


Models guided by data

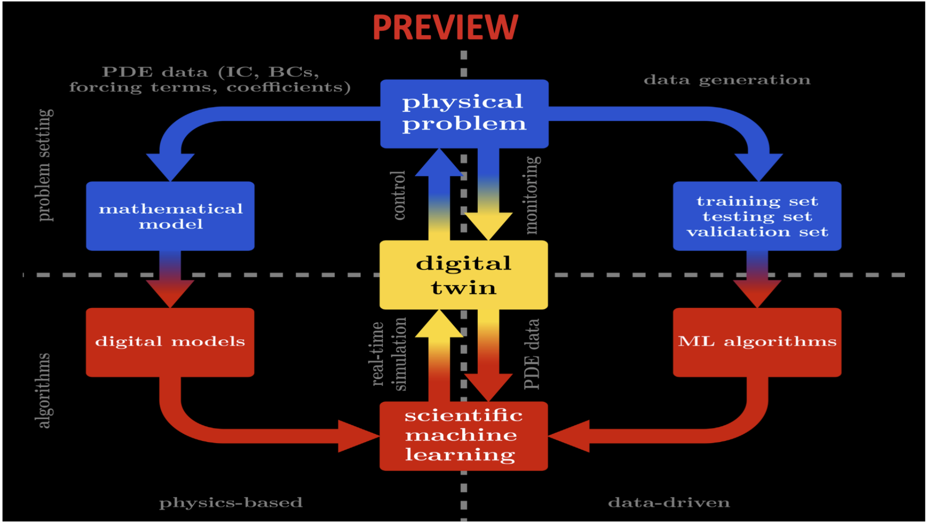
Available measurements $\longleftrightarrow$ model fitting

Neural networks :





The best of both worlds: scientific machine learning



[1]

[1] From the course of Alfio Quarteroni

2. Errors for classical models deployment: discretization

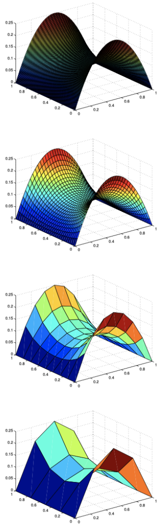
k time step, h space step, $r = \frac{k}{h^2}$
u_j^n: solution approximation at time instant n and spatial location j

$$\frac{u_j^{n+1} - u_j^n}{k} = \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{h^2}$$

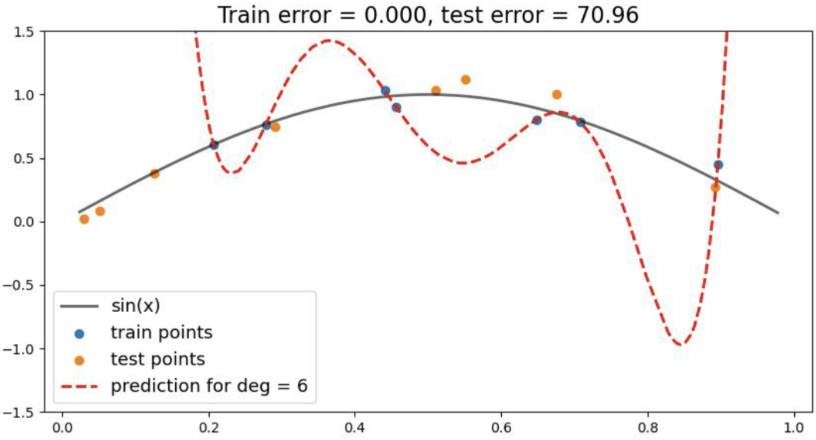
$$\Rightarrow (1+2r)u_j^{n+1} - ru_{j-1}^{n+1} - ru_{j+1}^{n+1} = u_j^n$$

$$\Rightarrow Au^{n+1} = f(u^n) \text{ Linear system}$$

Small k, h: large linear system $\rightarrow$ accurate ☺ but expensive ☹



2. Errors for modern models deployment: generalization



3. Rounding errors

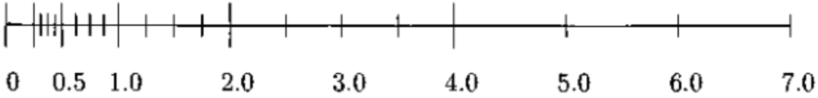
Floating-point numbers are represented by

$$x = \pm m \times \beta^{e-t}, \quad m \in [\beta^{t-1}, \beta^t - 1]$$

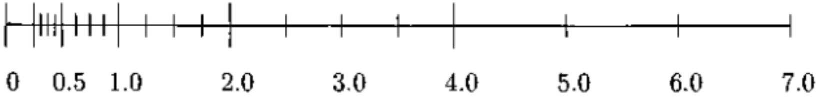
where

- β is the base (usually 2)
- t is the precision
- the sign $\pm$ is encoded on 1 bit
- the mantissa m is a t-bit integer with leading bit 1 (normalized)
- e is the exponent, satisfying $e \in [e_{\min}, e_{\max}]$

Example: $\beta = 2, \quad t = 3, \quad [e_{\min}, e_{\max}] = [-1, 3]$



Example: $\beta = 2, \quad t = 3, \quad [e_{\min}, e_{\max}] = [-1, 3]$



The unit roundoff $u = \beta^{1-t}/2$ ($= 2^{-t}$ in base 2) determines the relative accuracy any number in the representable range can be approximated with:

If $x \in \mathbb{R}$ belongs to $[e_{\min}, e_{\max}]$, then $\mathrm{fl}(x) = x(1+\delta), \quad |\delta| \le u$

Moreover the **standard model of arithmetic** is

$\mathrm{fl}(x \mathrm{\ op\ } y) = (x \mathrm{\ op\ } y)(1+\delta), \quad |\delta| \le u$, for $\mathrm{op} \in \{+, -, \times, \div\}$

Double and single precision

	Number of bits			Range	Unit roundoff u
	Sign	Mantissa	Exponent		
fp64	1	53	11	$10^{\pm 308}$	1×10^{-16}
fp32	1	24	8	$10^{\pm 38}$	6×10^{-8}

Double (fp64) and single (fp32) precision both widely supported in hardware

Even though 10^{-16} is tiny, rounding errors accumulate:
n operations $\Rightarrow$ n roundings errors!
Bounds of order nu can become problematic with large scale computations

Let's unpack the title (3/3)

Harnessing inexactness
in scientific computing

Let's unpack the title (3/3)

Harnessing inexactness
in scientific computing

What do we mean by harnessing inexactness ?

Approximate computing

Conclusion: today's computing is already approximate!
Since errors are part of scientific computing, let's embrace them: how can we harness inexactness?

4. **Approximation errors**
   - Rounding errors from use of low precision arithmetic
   - Compression/sparsification errors
   - Randomization
   - Errors from unstable algorithms
   - ...

Lower precisions: an opportunity

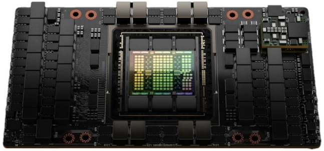
		number of bits			
		signif. (t)	exp.	range	$u = 2^{-t}$
fp128	quadruple	113	15	$10^{\pm 4932}$	1×10^{-34}
fp64	double	53	11	$10^{\pm 308}$	1×10^{-16}
fp32	single	24	8	$10^{\pm 38}$	6×10^{-8}
fp16	half	11	5	$10^{\pm 5}$	5×10^{-4}
bfloat16		8	8	$10^{\pm 38}$	4×10^{-3}
fp8 (e4m3)	quarter	4	4	$10^{\pm 2}$	6×10^{-2}
fp8 (e5m2)		3	5	$10^{\pm 5}$	1×10^{-1}

- **Great benefits:**
  - Reduced storage, data movement, and communications
  - Increased speed thanks to increasing hardware support
  - Reduced energy consumption
- However, **low precision** $\equiv$ **low accuracy**

Lower precisions: a necessity?

Peak performance (TFLOPS)

	Pascal 2016	Volta 2018	Ampere 2020	Hopper 2022	Blackwell 2025
fp64	5	8	20	67	40
fp32	10	16	20	67	80
tfloat32	--	--	160	495	2,200
fp16/bfloat16	20	125	320	990	4,500
fp8	--	--	--	2,000	9,000
fp4	--	--	--	--	18,000



NVIDIA Hopper (H100) GPU

fp64/fp16 speed ratio:

- Hopper (2022): 15×
- Blackwell (2025): 112×

Pros and cons of low precisions

- ☺ Storage, data movement and communications are all proportional to total number of bits (mantissa + exponent)
  lower precision ⇒ lighter computations

- ☺ Speed of computations *at least* proportional
  - on most architectures, fp32 is 2× faster than fp64
  - on some architectures, fp16/bfloat16 can be 10-100× faster than fp32

  lower precision ⇒ faster computations

- ☺ Power consumption is proportional to the square of the number of mantissa bits
  - fp16 (11 bits) consumes 5× less energy than fp32 (24 bits)
  - bfloat16 (8 bits) consumes 9× less energy than fp32

  lower precision ⇒ greener computations

- ☹ Errors are proportional to the unit roundoff
  lower precision ⇒ lower accuracy

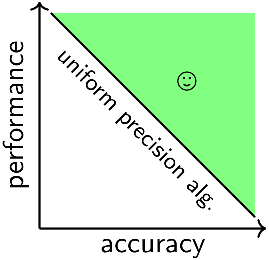
Mixed precision algorithms

Mix several precisions in the same code with the goal of

- Getting the **performance benefits of low precisions**
- While preserving the **accuracy and stability of the high precision**

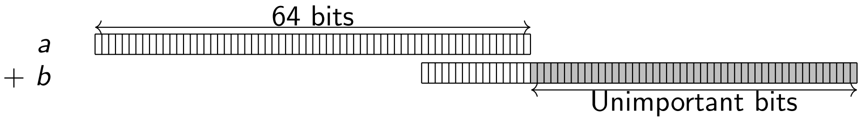
- Want to use as much as possible low precisions, as little as possible high precisions
- **Goal** (compared with uniform precision)
  - improve accuracy for a small extra cost
  - improve performance at a controlled loss of accuracy

$\Rightarrow$ in both cases, we seek an **improved performance–accuracy tradeoff**



Adaptive precision algorithms

- Adaptive precision algorithms: a subclass of mixed precision algorithms which dynamically adapt the precision of each variable/instruction depending on the data
- Example:



⇒ **Opportunity for mixed precision:** adapt the precisions to the data at hand by storing and computing "less important" (which usually means smaller) data in lower precision

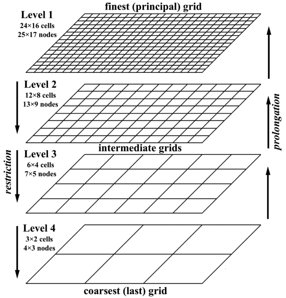
Accuracy in iterative processes

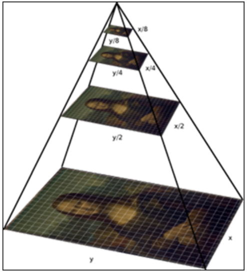
- Many of the applications require iterative processes to be solved
- High accuracy $\leftrightarrow$ high cost
- Do we need the same accuracy along all the process?

$\rightarrow$ several ways of reducing the accuracy

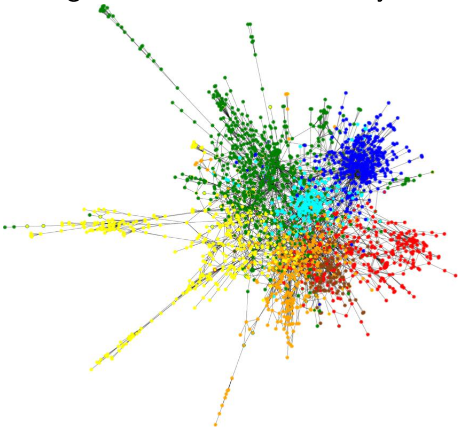
Exploit the structure to build approximated subproblems of reduced size





- How to built the subproblems?
- When to use them?

Large datasets $\rightarrow$ redundancy $\rightarrow$ subsampled methods



- How to select subsampled sets?
- How large sets?
- How to vary this size?

Structured matrices

Examples of **matrices with a special structure**:

- Sparse matrices
- Numerically sparse matrices
- Low-rank matrices
- Data sparse matrices
- ...

Two challenges:

- Given an unstructured matrix, **find a good structured approximation**
- Given a structured matrix, **develop specialized algorithms to take advantage of its structure**

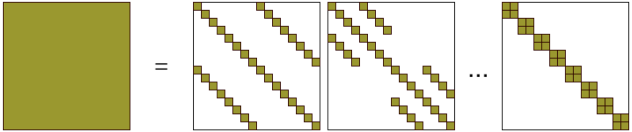
Sparse matrices

$$\begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 7 & 0 & 3 \\ -2 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 \end{pmatrix} \Rightarrow \begin{array}{ccc} \mathit{ROW\_IND} & \mathit{COL\_IND} & \mathit{VAL} \\ \begin{bmatrix} 1 \\ 2 \\ 4 \\ 5 \end{bmatrix} & \begin{bmatrix} 1 \\ 2 \\ 4 \\ 1 \\ 3 \end{bmatrix} & \begin{bmatrix} 4 \\ 7 \\ 3 \\ -2 \\ 5 \end{bmatrix} \end{array}$$

Storage cost:

- Dense: n^2 floating-point entries
- Sparse: nnz floating-point entries and $nnz + n$ integer indices

Butterfly matrices

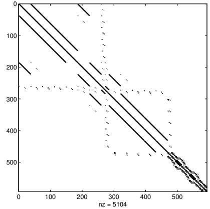


- Extremely sparse matrices
- Highly expressive
- Structured sparsity (allows for GPU acceleration)
- Basis for fast transforms: Hadamard, Fourier...
- Used also in machine learning to compress neural networks or to speed up training (2x faster)

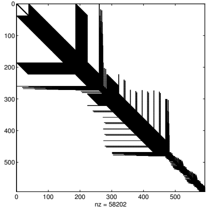
Sparse Gaussian elimination

Gaussian elimination (LU factorization): $a_{ij} \leftarrow a_{ij} - a_{ik}a_{kj}$
$\Rightarrow$ a_{ij} becomes nonzero if a_{ik} and a_{kj} are nonzero: fill-in
Example: dwt_592.rua, structural computing on a submarine.

Original matrix



Factorized matrix



Computational costs heavily dependent on matrix structure and permutation. For regular 3D problems (PDE discretized on a cube):

- LU flops: $O(n^3) \rightarrow O(n^2)$
- LU storage: $O(n^2) \rightarrow O(n^{4/3})$

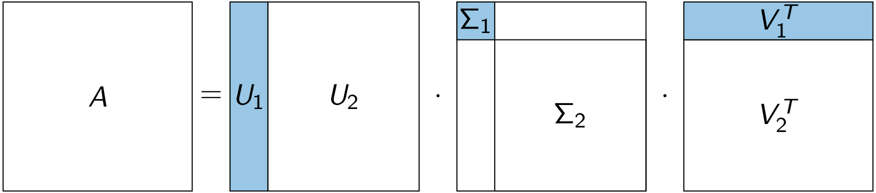
Incomplete factorizations

- "Numerically sparse" matrix: a matrix that becomes sparse by dropping its entries smaller than a threshold ϵ (in absolute value)
- Incomplete factorizations: drop entries $< \epsilon$ from LU factors
- Alternatively, do not update $a_{ij} \leftarrow a_{ij} - a_{ik}a_{kj}$ if a_{ij} is zero (i.e., enforce same sparsity pattern for LU as for A)
- Can work well for some matrices, but lacks subtility: matrices are usually only numerically sparse for large ϵ

Low-rank matrices

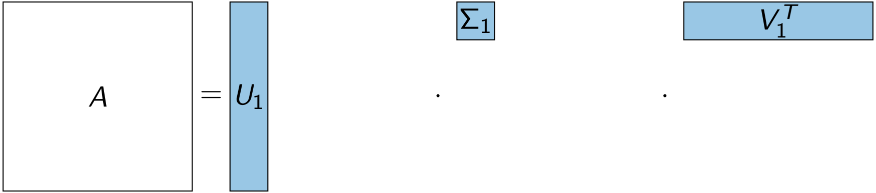
$$A = U \cdot \Sigma \cdot V^T$$

Low-rank matrices



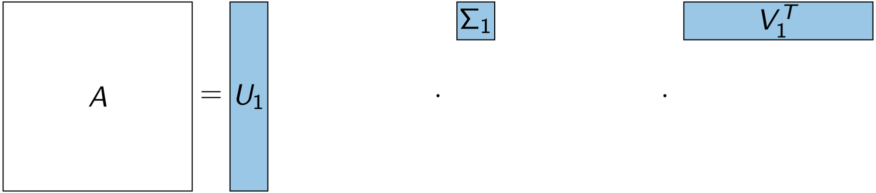
$A = U_1 \Sigma_1 V_1^T + U_2 \Sigma_2 V_2^T$ with $\Sigma_1(k,k) = \sigma_k > \varepsilon,\ \Sigma_2(1,1) = \sigma_{k+1} \leq \varepsilon$

Low-rank matrices



$$A = U_1 \Sigma_1 V_1^T + U_2 \Sigma_2 V_2^T \quad \text{with} \quad \Sigma_1(k,k) = \sigma_k > \varepsilon, \; \Sigma_2(1,1) = \sigma_{k+1} \leq \varepsilon$$

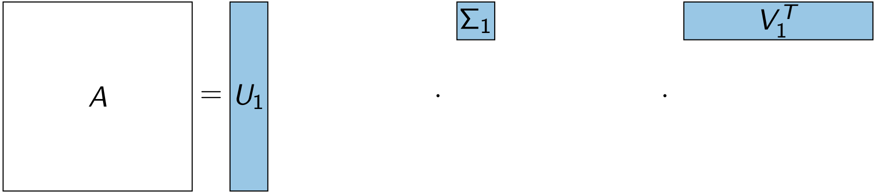
If $\tilde{A} = U_1 \Sigma_1 V_1^T$ then $\|A - \tilde{A}\|_2 = \|U_2 \Sigma_2 V_2^T\|_2 = \sigma_{k+1} \leq \varepsilon$



$A = U_1 \Sigma_1 V_1^T + U_2 \Sigma_2 V_2^T \quad$ with $\quad \Sigma_1(k,k) = \sigma_k > \varepsilon,\ \Sigma_2(1,1) = \sigma_{k+1} \leq \varepsilon$

If $\tilde{A} = U_1 \Sigma_1 V_1^T \quad$ then $\quad \|A - \tilde{A}\|_2 = \|U_2 \Sigma_2 V_2^T\|_2 = \sigma_{k+1} \leq \varepsilon$

If $k < mn/(m+n)$, $\tilde{A}$ requires less storage than $A \Rightarrow$ low-rank matrix.



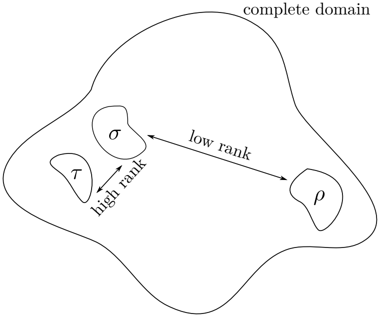
$A = U_1 \Sigma_1 V_1^T + U_2 \Sigma_2 V_2^T$ with $\Sigma_1(k,k) = \sigma_k > \varepsilon,\ \Sigma_2(1,1) = \sigma_{k+1} \leq \varepsilon$

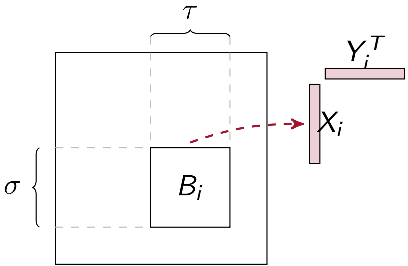
If $\tilde{A} = U_1 \Sigma_1 V_1^T$ then $\|A - \tilde{A}\|_2 = \|U_2 \Sigma_2 V_2^T\|_2 = \sigma_{k+1} \leq \varepsilon$

If $k < mn/(m+n)$, $\tilde{A}$ requires less storage than $A \Rightarrow$ low-rank matrix.

SVD cost: $O(mn \min(m,n))$ flops $\Rightarrow$ too expensive for large matrices. Other (suboptimal) methods are used in practice.

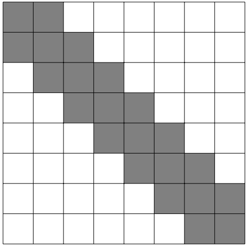
Block low-rank (BLR) approximations





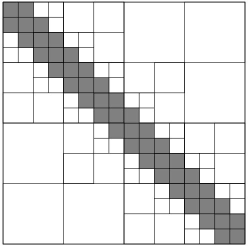
Block low-rank (BLR) approximations

Many different block partitionings possible



BLR matrix

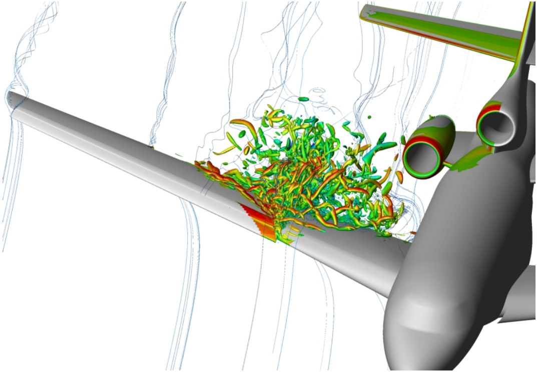
- Simple, flat structure
- Superlinear complexity



$\mathcal{H}$-matrix

- Complex, hierarchical structure
- Near-optimal loglinear complexity

Let's unpack the illustration



Unpacking the course

Approximate computing, *vue d'avion*

Ad break

Practical organization

Vue d'avion: the scientific computing stack

Applications
Methods
Software
Hardware

- In an ideal world, each layer would be independent.

Vue d'avion: the scientific computing stack

Applications	ideally everything below is a black box for end-users
Methods	mathematical pen & paper algorithms
Software	libraries provide standard building blocks (BLAS, LAPACK)
Hardware	should follow some standard rules (e.g., IEEE 754) vendors provide optimized software implementations

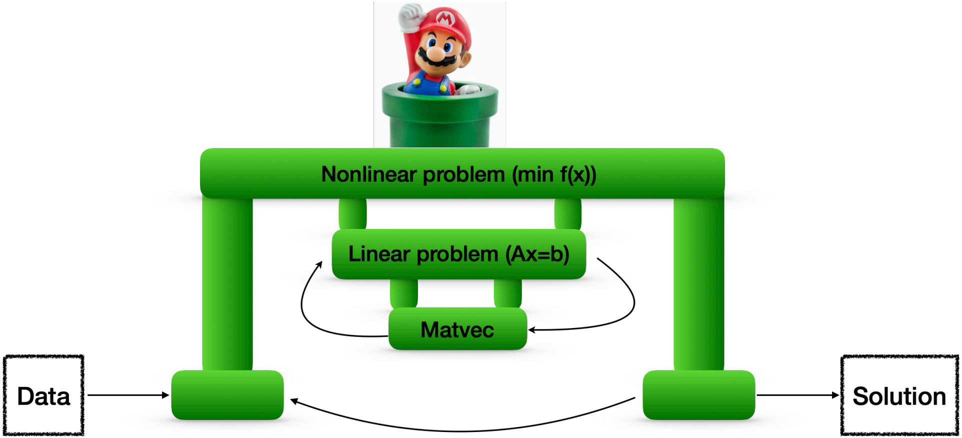
- In an ideal world, each layer would be independent.
- With traditional scientific computing, this separation mostly holds.

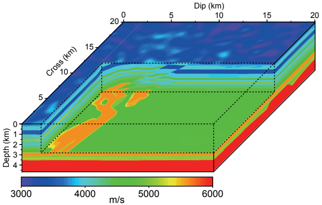
Vue d'avion: the scientific computing stack

Applications	performance–accuracy tradeoffs can only be meaningfully assessed on an application-by-application basis
Methods	application-specific methods (PINNs, compression) hardware-aware methods (mixed precision, randomization, communication avoidance)
Software	
Hardware	specialized hardware (TPUs, tensor cores, . . .), non-standard arithmetics (bfloat16, fp8 and lower, . . .)

- In an ideal world, each layer would be independent.
- With traditional scientific computing, this separation mostly holds.
- With approximate computing, things become much more interconnected!

Vue d'avion: the scientific computing pipeline





(3D EAGE/SEG overthrust model)

(credits: SEISCOPE project)

$\Longrightarrow$

Frequency domain FWI (Full-Wave Inversion)

Helmholtz equations

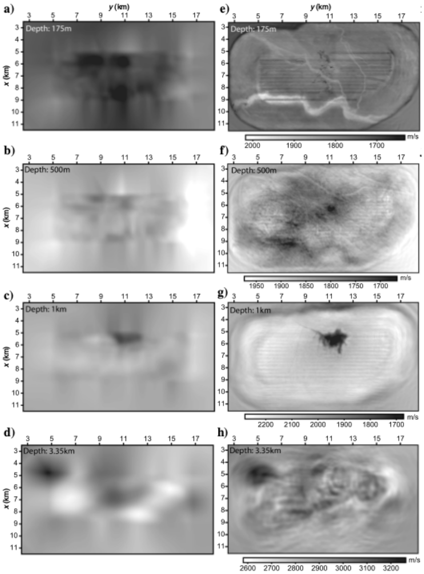
Complex Unsym. sparse matrix **A**

Multiple (very) sparse **B**

Required accuracy $< 10^{-4}$

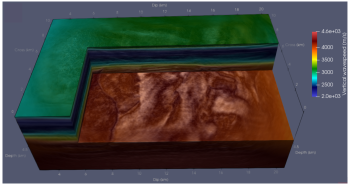
freq	flops LU	Factor Storage	Peak memory
2 Hz	9.0E+11	3 GB	4 GB
4 Hz	1.6E+13	22 GB	25 GB
8 Hz	5.8E+14	247 GB	283 GB
10 Hz	2.7E+15	728 GB	984 GB

Higher frequency leads to refined model


a)
y (km)
x (km)
Depth: 175m
e)
y (km)
Depth: 175m
m/s
2000 1900 1800 1700
b)
Depth: 500m
f)
Depth: 500m
m/s
1950 1900 1850 1800 1750 1700
c)
Depth: 1km
g)
Depth: 1km
m/s
2200 2100 2000 1900 1800 1700
d)
Depth: 3.35km
h)
Depth: 3.35km
m/s
2600 2700 2800 2900 3000 3100 3200

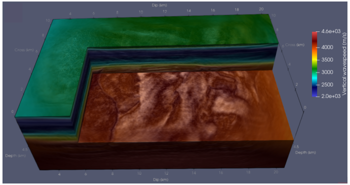
Seismic imaging in geophysics

- Adastra MUMPS4FWI project led by WIND team
- Application: Gorgon Model, reservoir 23km x 11km x 6.5km, grid size 15m, Helmholtz equation, 25-Hz
- Complex matrix, 531 Million dofs, storage(A)=220 GBytes;
- FR cost: flops for one LU factorization$= 2.6 \times 10^{18}$;
  Estimated storage for LU factors$=$ 73 TBytes



(25-Hz Gorgon FWI velocity model)

- Adastra MUMPS4FWI project led by WIND team
- Application: Gorgon Model, reservoir 23km x 11km x 6.5km, grid size 15m, Helmholtz equation, 25-Hz
- Complex matrix, 531 Million dofs, storage(A)=220 GBytes;
- FR cost: flops for one LU factorization= 2.6×10^{18};
  Estimated storage for LU factors= 73 TBytes



(25-Hz Gorgon FWI velocity model)

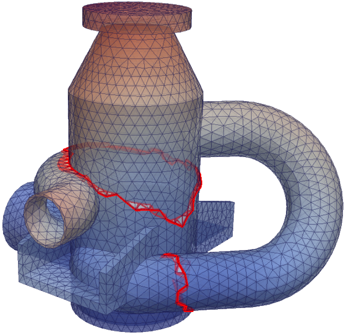
FR (Full-Rank); BLR with $\varepsilon = 10^{-5}$; 48 000 cores (500 MPI $\times$ 96 threads/MPI)

FR: fp32; Mixed precision BLR: 3 precisions (32bits, 24bits, 16bits) for storage

LU size (TBytes)			Flops		Time BLR + Mixed (sec)			Scaled Resid.
FR	BLR	+mixed	FR	BLR+mixed	Analysis	Facto	Solve	BLR+mixed
73	34	26	2.6×10^{18}	0.5×10^{18}	446	5500	27	7×10^{-4}

in practice: hundreds to thousands of Solve steps (sparse right hand sides (sources))

Structural mechanics



A RIS pump (circuit d'injection de sécurité) under internal pressure

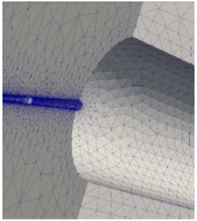
Real sym. **indefinite** sparse matrix **A**

One dense right-hand side **b**

Required accuracy $> 10^{-9}$

n	nnz	flops LU	LU Storage
5.4E+6	2.1E+8	1.8E+13	56 GB

Number of delayed pivots $=$ 79k



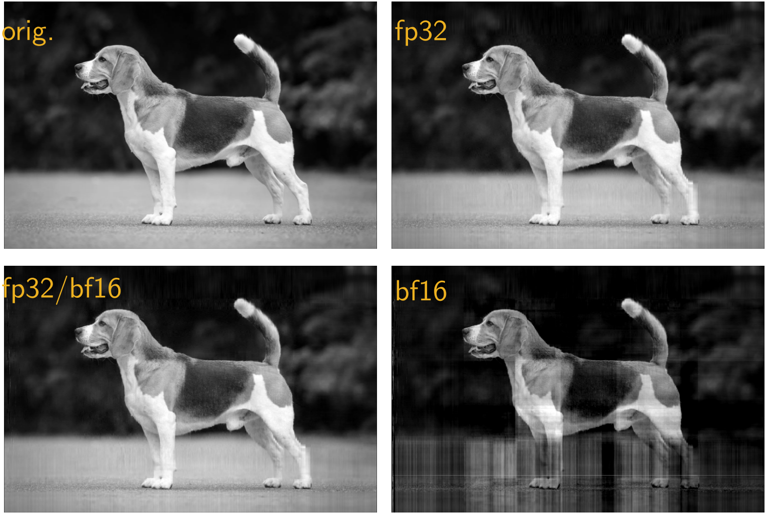
(from code_aster)

- thmgaz matrix ($n = 5M$)
  - multi-physics (thermo-hydro-mechanics)
  - 2 MPI $\times$ 18 threads
  - MUMPS solver Amestoy, Buttari, L'Excellent, M. (2019)

	Facto. time (s)	Memory (GB)
Full-rank double	98	192
BLR ($\varepsilon = 10^{-8}$) double	81	131
Full-rank single + LU-IR	65	98
BLR ($\varepsilon = 10^{-8}$) single + LU-IR	59	67
BLR ($\varepsilon = 10^{-6}$) single + GMRES-IR	71	61

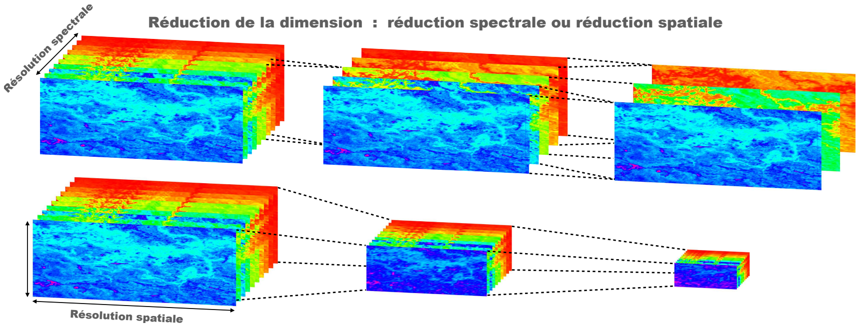
Amestoy, Buttari, Higham, L'Excellent, M., Vieublé (2023)

Image compression



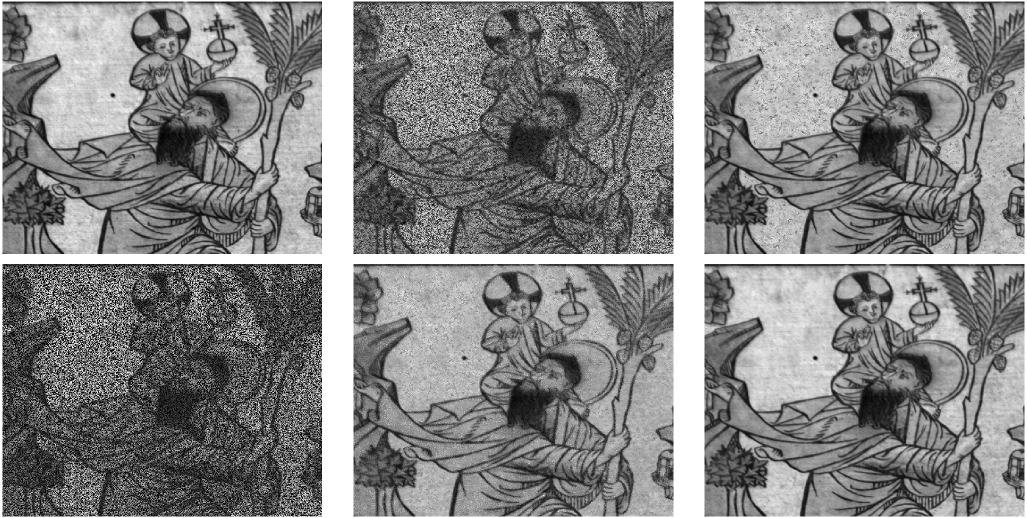
With $\varepsilon = 0.04$ the rank is 191 but only 13 steps are done in fp32 and the rest in bf16 (original size is 1057×1600)

The goal of hyperspectral imaging is to obtain the spectrum for each pixel in the image of a scene, with the purpose of finding objects, identifying materials, or detecting processes



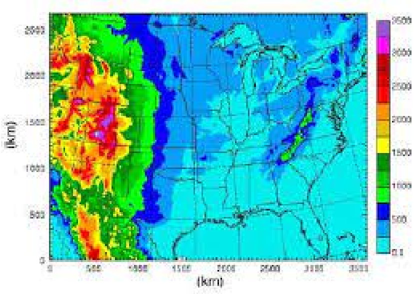
Spectral image restoration

$\bar{x}$ (SNR) $x_{2,\mathtt{FISTA}}$ (7 dB) $x_{\mathrm{end},\mathtt{FISTA}}$ (21 dB)



z (3 dB) $\mathrm{x}_{2,\mathtt{IML\ FISTA}}$ (19 dB) $\mathrm{x}_{\mathrm{end},\mathtt{IML\ FISTA}}$ (35 dB)

Weather forecasting



$$\min_{x \in \mathbb{R}^n} \frac{1}{2}\|x - x_b\|^2_{B^{-1}} + \frac{1}{2}\sum_{j=0}^{N_t} \|H_j(x(t_j)) - y_j\|^2_{R_j^{-1}}$$

Reduction up to 60% with subsampling strategies

Navier-Stokes equation on Ω

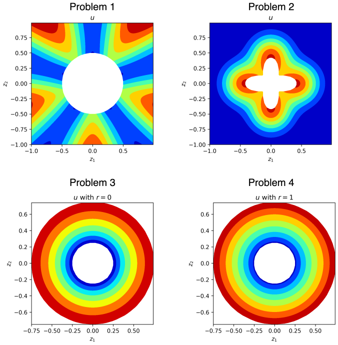
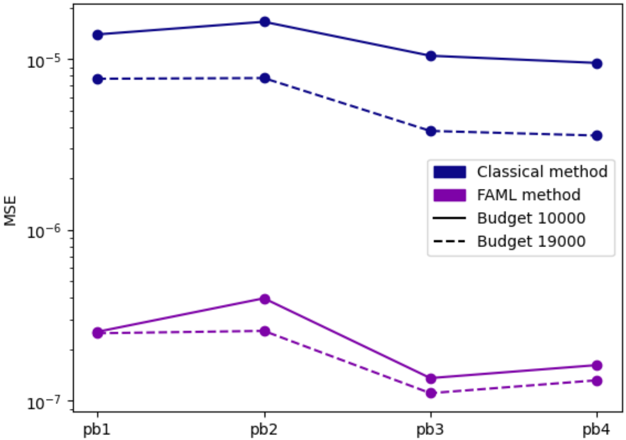


Figure: Ω



Some fundamental research problems

These case studies illustrate the high stakes behind approximate computing, and industrial interests that it attracts…

…however, approximate computing also raises **fundamental research problems**

- What is the **impact of approximations?** How to measure and control it?
  - Rounding error bounds, convergence rate analysis, attainable accuracy, …
- What **kind of problems/data** are amenable to approximations?
  - What happens if we try approximations on a non-amenable case? A priori estimators, a posteriori checks, complexity bounds…
- How can we efficiently **translate approximations into performance**?
  - Performance optimization, parallel scalability, communication avoidance, …

National projects involving approximate computing

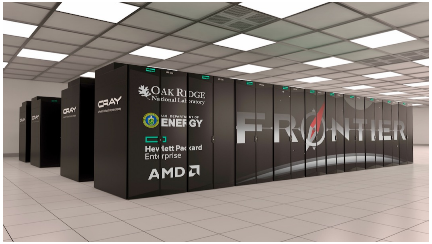
The combination of industrial stakes and fundamental research problems has led to the creation of several large national projects:

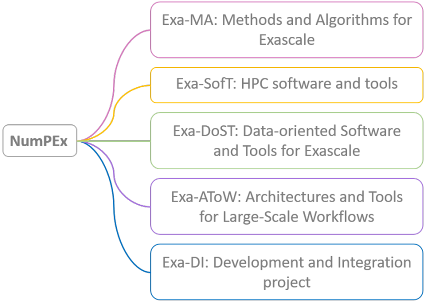
PROGRAMME DE RECHERCHE
NUMÉRIQUE POUR L'EXASCALE

PROGRAMME DE RECHERCHE
INTELLIGENCE ARTIFICIELLE

PEPR NumPEx

- **Exascale** computing: ability to perform 10^{18} floating-point operations per second. Such capability will allow for technological breakthroughs in all societal domains.
- June 2022: Frontier achieves 1.1 ExaFlops/s and becomes the world's first exascale computer
- Exascale computers expected soon in Europe and France. Are we ready to exploit them?
- Goals of NumPEx: designing and developing software components that will equip future exascale machines





PEPR NumPEx (ExaMA)

Exascale computing offers great promises. . . but do we have methods able to exploit such computational power?!

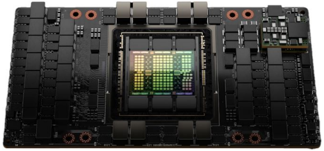
Key challenges:

- Huge amounts of parallelism/concurrency
- High heterogeneity in the computing units: CPUs, GPUs, other accelerators
- Large gap between speed of computations and communications
- Expensive power consumption

Possible solutions:

- Scalable, parallel methods
- Mixed precision methods
- Randomized methods
- Communication-avoiding methods

- Explore the mathematical and algorithmic foundations of sparse deep learning (sparse factorization, dimension-reduction techniques, quantization principles)
- Optimally combine traditional learning with knowledge in symmetries, prior probabilistic models, and representation learning, to reduce the dimension of models and the amount of data required.



NVIDIA Hopper GPU

NVIDIA currently has a monopoly on hardware specialized for AI workloads with GPU tensor cores. Monopolies are never a good idea, especially when held by a foreign, private company.

- Goal of the project: design a sovereign alternative for AI hardware, from the hardware itself to the software stack on top of it
- What features of the hardware (matrix multiply–accumulate, stochastic rounding, extended precision ...) should we aim for and how can we leverage them if available?
- What is the impact of such features on AI applications, in particular the training and inference of neural networks?

Unpacking the course

Approximate computing, *vue d'avion*

Ad break

Practical organization

Internship positions, i.e., open research problems :-)

Title	Location
Adaptive precision iterative solvers	LIP6 (Paris)
Domain decomposition block low-rank preconditioners	ONERA (Paris)
Composable error analyses for linear algebra	Inria Bordeaux
Probabilistic error analysis of matrix factorizations	LIP6 (Paris)
Certified fast transforms	LIP (ENS Lyon)
Error analysis of fast transforms for deep learning	LIP (ENS Lyon)
Optimal quantization of neural networks	LIP (ENS Lyon)
Quantized butterfly matrices	LIP (ENS Lyon)
Mixed precision domain decomposition training methods	IRIT (Toulouse)

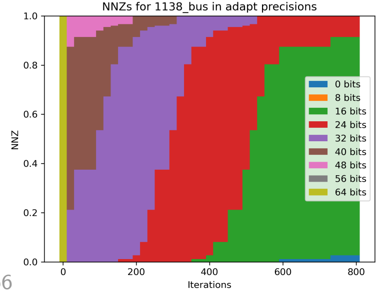
See course webpage for details!

https://perso.ens-lyon.fr/elisa.riccietti/stages

Internship: adaptive precision iterative solvers

- Goal: efficiently solve large sparse linear systems with mixed precision iterative methods
- **Lecture 2**: adaptive precision matrix–vector product: vary the precisions across different coefficients of the matrix, based on their magnitude
- **Lecture 9**: relaxed Krylov methods: decrease the precision across the iterations, based on the residual decrease

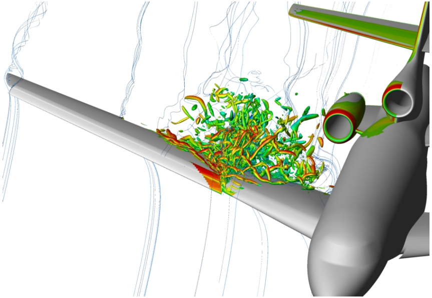
$\Rightarrow$ combine both!



- Can these theoretical reductions be translated into actual performance gains?
- Study on large scale problems and parallel computers needed

$\Rightarrow$ See here for details

Internship: domain decomposition block low-rank preconditioners

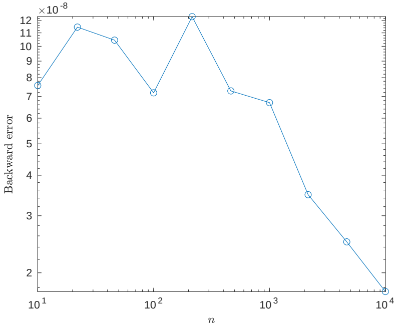


ONERA problems: computational fluid dynamics for aerospatial applications
Goal: solve very large sparse linear systems. Direct methods (**Lecture 6**) too costly (OOM), iterative ones (**Lecture 9**) do not converge $\Rightarrow$ need lightweiht yet robust preconditioners

- Current solution: **domain decomposition methods**, with either ILU0 or exact LU local solvers
  - Exact LU $\Rightarrow$ expensive, memory consumption is the limiting factor
  - ILU0 $\Rightarrow$ very slow convergence
- Goal 1: add **block low-rank LU** (**Lecture 14**) as an alternative for local solvers, with a better tradeoff between memory compression and preconditioner quality
- Goal 2: develop **adaptive strategies** that use different solvers (ILU, BLR, exact) for different domains

Internship: probabilistic error analysis of matrix factorizations

- Traditional worst-case error bounds for linear algebra computations involving matrices of order n in precision u are proportional to nu
- Very unsatisfactory if n and/or u are large
- **Lecture 4**: can reduce nu to $\sqrt{n}u$ thanks to statistical effets / stochastic rounding
- ... yet, some computations remain inexplicably even more accurate



- Error for matrix factorization (here, $A = LL^T$) does not grow with n at all! Why?
- Impact on TOP500? Accuracy criterion to accept a given benchmark

$\Rightarrow$ See here for details

Internship: optimal quantization of neural networks

Previous work

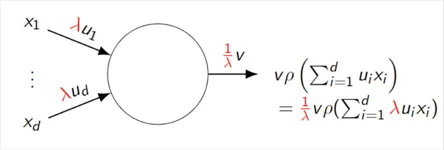
- **Optimal quantization of rank-one matrices** by exploiting **rescaling symmetries**

$$\min_{\hat{x}\in\mathbb{F}^m,\hat{y}\in\mathbb{F}^n} \|xy^T - \hat{x}\hat{y}^T\|$$

$$\boxed{xy^T = (\lambda x)\left(\frac{y}{\lambda}\right)^T}$$

Future work

- Similar symmetries can be found in **ReLU neural networks:** theoretically founded **quantization schemes?**

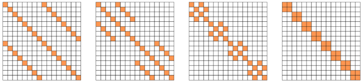


$$\rho(t) = \max(0,t), \lambda > 0$$

Internship: quantized butterfly matrices

<u>Previous work</u>

- Approximation of networks weight matrices by butterfly matrices



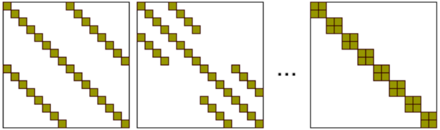
B_1 B_2 B_3 B_4

- Heuristic strategy for the quantization of **butterfly factorization** with **30% gain** in storage wrt RTN

$$\min_{\hat{B}_i \in \mathbb{F}^{n\times n}} \|A - \hat{B}_1 \ldots \hat{B}_L\|$$

<u>Future work</u>

- Study of **approximation of weight matrices** in neural networks by **quantized butterflies**?



$\hat{B}_1$ $\hat{B}_2$ ... $\hat{B}_L$

Unpacking the course

Approximate computing, *vue d'avion*

Ad break

Practical organization

Outline

Number	Date	Time	Lecturer	Title
1	Nov 18	10:00	TM+ER	Introduction
2	Nov 22	13:30	TM	Summation methods
3	Nov 25	10:00	ER	Stochastic optimization methods
4	Nov 29	13:30	TM	Probabilistic error analysis
5	Dec 2	10:00	ER	Multigrid and multilevel methods
6	Dec 6	13:30	TM	Direct linear solvers
7	Dec 9	10:00		no lecture
8	Dec 13	13:30	ER	Image restoration
9	Dec 16	10:00	TM	Iterative linear solvers
10	Dec 20	13:30	ER	PINNs: physics informed neural networks
11	Jan 6	10:00	TM	Low-rank approximations
12	Jan 10	13:30	ER	Neural networks and low precision arithmetic
13	Jan 13	10:00		no lecture
14	Jan 17	13:30	TM	Block low-rank matrices
15	Jan 20	10:00	ER	Sparse approximation
16	Jan 24	13:30	TM	GPU numerical computing
17	Jan 27	10:00	TM+ER	Bonus session (TBD)
18	Jan 31	13:30	TM+ER	Oral evaluation

Evaluation

Evaluation will be composed of two parts:

- Practical exercises at home/between lectures $\rightarrow$ mark M_1
- Oral evaluation $\rightarrow$ mark M_2

$\rightarrow$ Final mark: $M = (M_1 + M_2)/2$

Evaluation: practical exercises

- Timeline:
  - Given at the end of some lectures
  - To be sent by email to the lecturer before the next lecture *by the same lecturer*
  - Corrected at the beginning of that lecture
  - Example: at the end of lecture 2 (summation), TM presents the exercise; solutions should be sent to TM before the beginning of the next lecture by TM, lecture 4 (probabilistic), at the beginning of which, TM will correct the exercise.
- Questions/help can be asked by email, we will *try* to respond if we can.
- Barème (scale):
  - A correct solution is worth 4 points; a non-trivial (but incorrect) attempt is worth 2 points.
  - There should be around 8 exercises (but you only need 5 exercises to get the full mark).
- The exercises will be in MATLAB (TM) and Python (ER).
  - MATLAB student licenses can be obtained for free here: https://fr.mathworks.com/academia/tah-portal/ens-lyon-31067144.html

Evaluation: oral presentation

- Goal: read a research article and present a *critical* commentary of it
- Oral presentation, with slides (beamer strongly recommended), duration TBD
- Presentation should not be restricted to a mere summary but should provide *critical commentary*: what is the originality/novelty of the work? Why is it important, and why now? What are its strengths? Its weaknesses or limitations? What perspectives does it open and what future work should be considered?
- You should be equipped with the skills to do so by the end of the course: we will not just present some cool methods but also go look under the hood and discuss all the difficulties that arise in practice
- **List of research articles with PDFs available on course webpage**

Theo Mary (CNRS)
theo.mary@lip6.fr
https://perso.lip6.fr/Theo.Mary/

Elisa Riccietti (ENS Lyon)
elisa.riccietti@ens-lyon.fr
https://perso.ens-lyon.fr/elisa.riccietti/

Slides available on course webpage

Thanks!
Questions?