CANUM 2022

14 June 2022

Solving linear systems efficiently using block low-rank compression in mixed precision

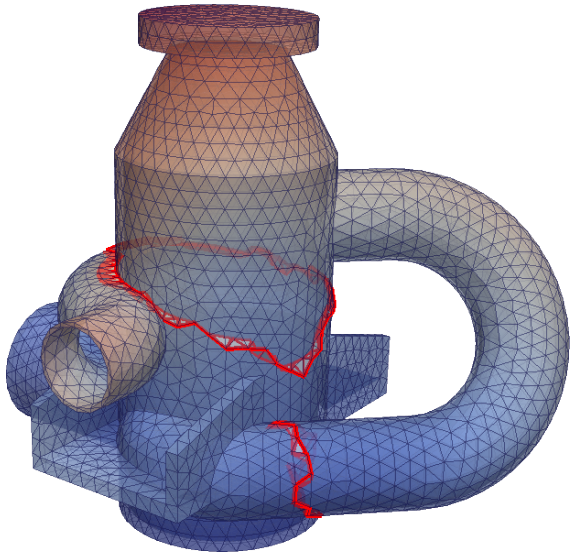
Matthieu Gerest
EDF R&D, LIP6 (CIFRE PhD)

Join work with:
Patrick Amestoy[1], Olivier Boiteau[2], Alfredo Buttari[3], Fabienne Jézéquel[4], Jean-Yves L'Excellent[1], Théo Mary[4]

[1]Mumps Technologies
[2]EDF R&D
[3]Université de Toulouse, CNRS, IRIT
[4]Sorbonne Université, CNRS, LIP6

Context: solving linear systems

- Objective: solving $Ax = b$
- Direct method: compute LU=A
- Often a bottleneck in terms of memory bandwith and computation time
- MUMPS: a multifrontal solver

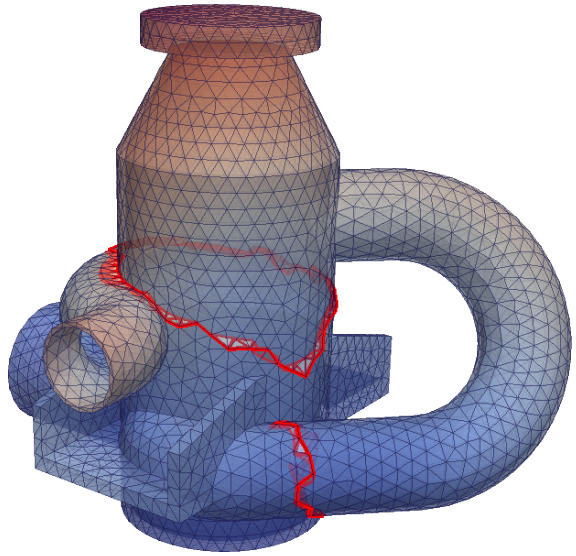


Example of application: a RIS pump under internal pressure

code_aster

Context: solving linear systems

- Objective: solving $Ax = b$
- Direct method: compute LU=A
- Often a bottleneck in terms of memory bandwith and computation time
- MUMPS: a multifrontal solver
- BLR compression: a method to reduce the complexity
- Objective: combine BLR compression and mixed precision

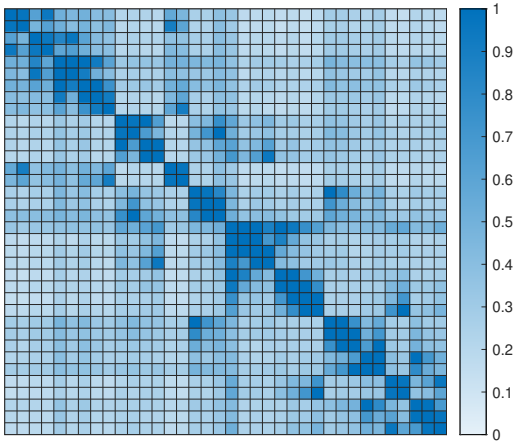


Example of application: a RIS pump under internal pressure

code_aster

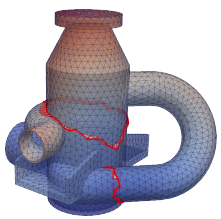
BLR compression

- Class of matrices: BLR matrices
- Off-diagonal blocks of A tend to contain less information (low numerical ranks)



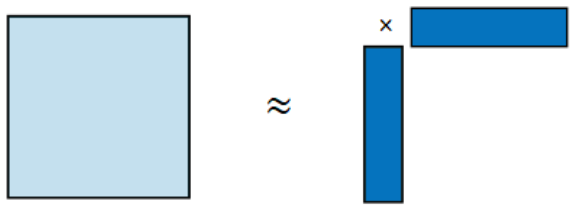
Example of BLR matrix (*perf009*, RIS pump)

Color scale: numerical ranks of the blocks for $\varepsilon = 10^{-10}$

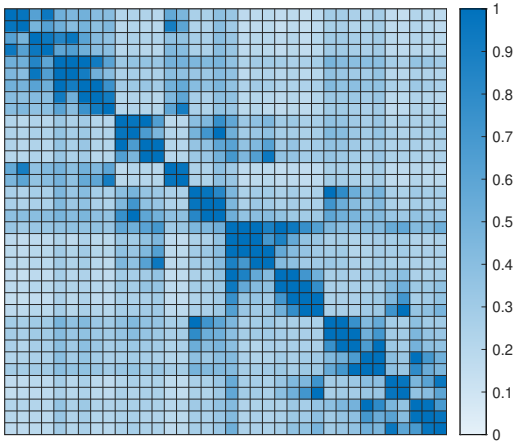


BLR compression

- Class of matrices: BLR matrices
- Off-diagonal blocks of A tend to contain less information (low numerical ranks)
- They can be replaced by their Low-Rank approximations

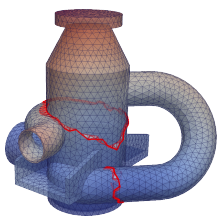


$$B \approx X \times Y^T$$



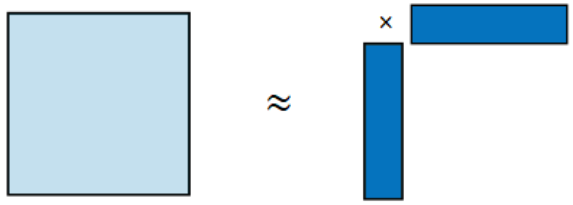
Example of BLR matrix (*perf009*, RIS pump)

Color scale: numerical ranks of the blocks for $\varepsilon = 10^{-10}$



BLR compression

- Class of matrices: BLR matrices
- Off-diagonal blocks of A tend to contain less information (low numerical ranks)
- They can be replaced by their Low-Rank approximations



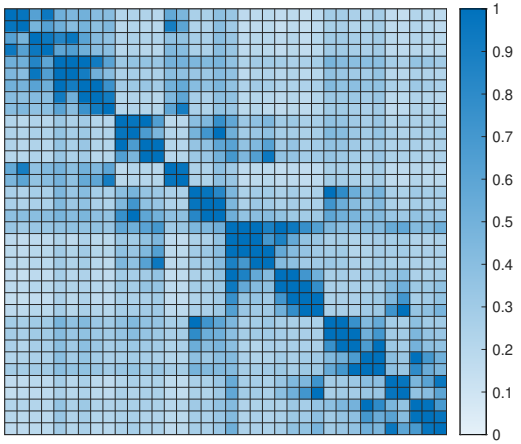
$$B \approx X \times Y^T$$

- The compression error is controlled by a threshold ε
- Larger ε

  $\Rightarrow$ fewer coefficients stored

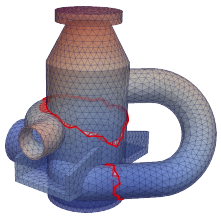
  $\Rightarrow$ fewer operations

  $\Rightarrow$ faster computations

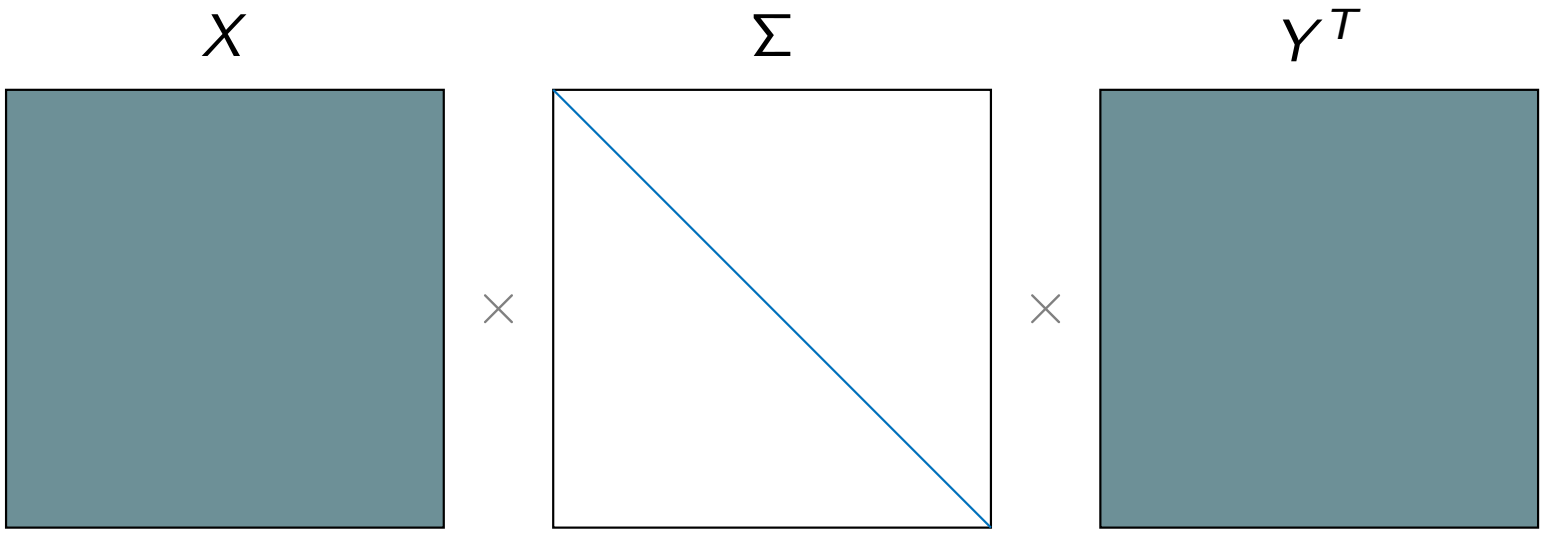


Example of BLR matrix (*perf009*, RIS pump)

Color scale: numerical ranks of the blocks for $\varepsilon = 10^{-10}$

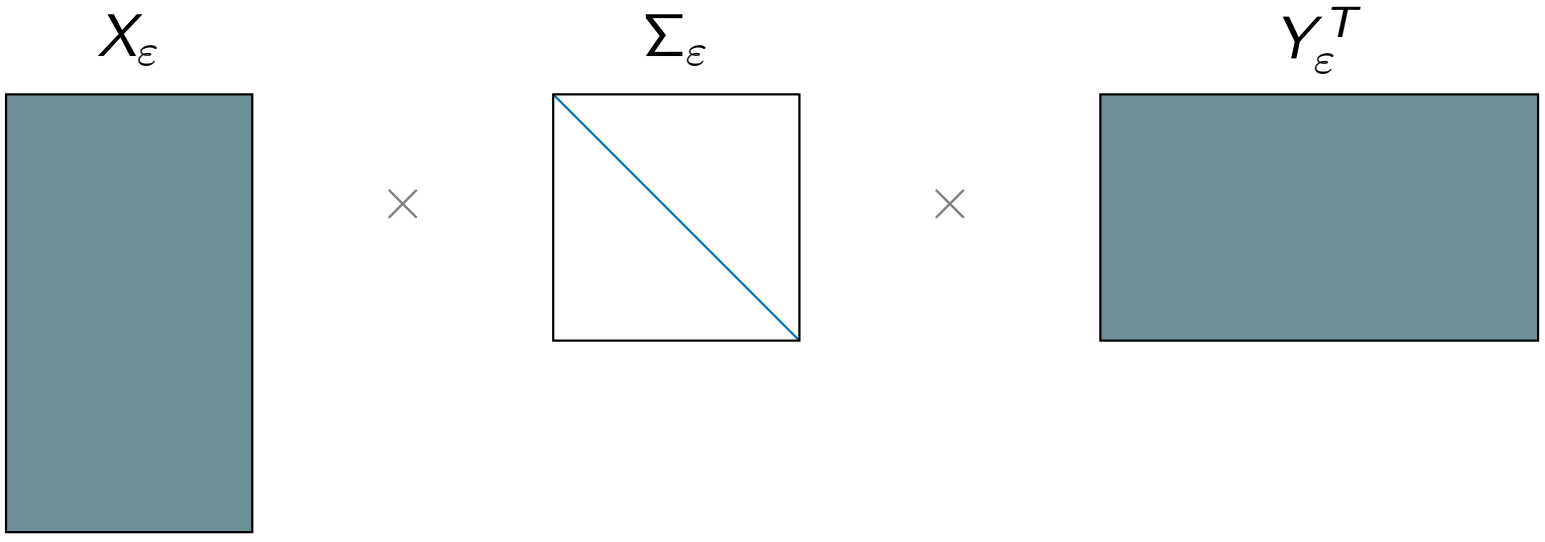


Mixed precision Low-Rank approximation



SVD: Singular Value Decomposition

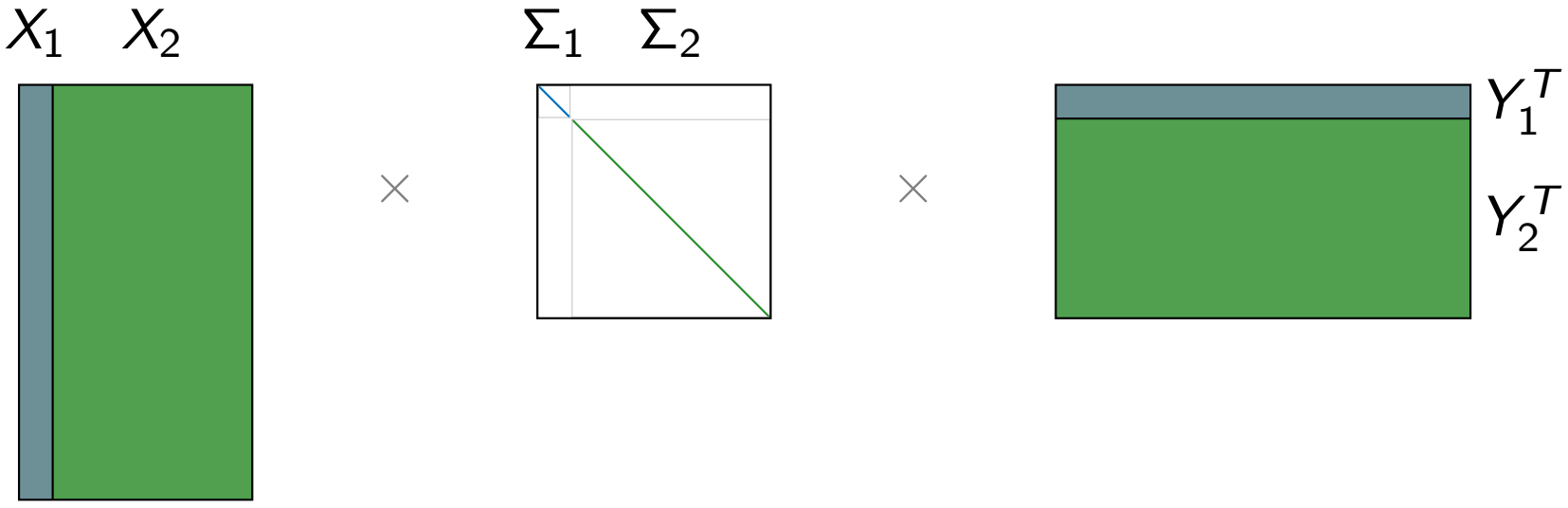
Mixed precision Low-Rank approximation



Truncated SVD

- $B = \sum_{k=1}^{r} x_k \sigma_k y_k^T$, with r such that the error satisfies
- $\|B - X_\varepsilon \Sigma_\varepsilon Y_\varepsilon\| \leq \varepsilon \|B\|$

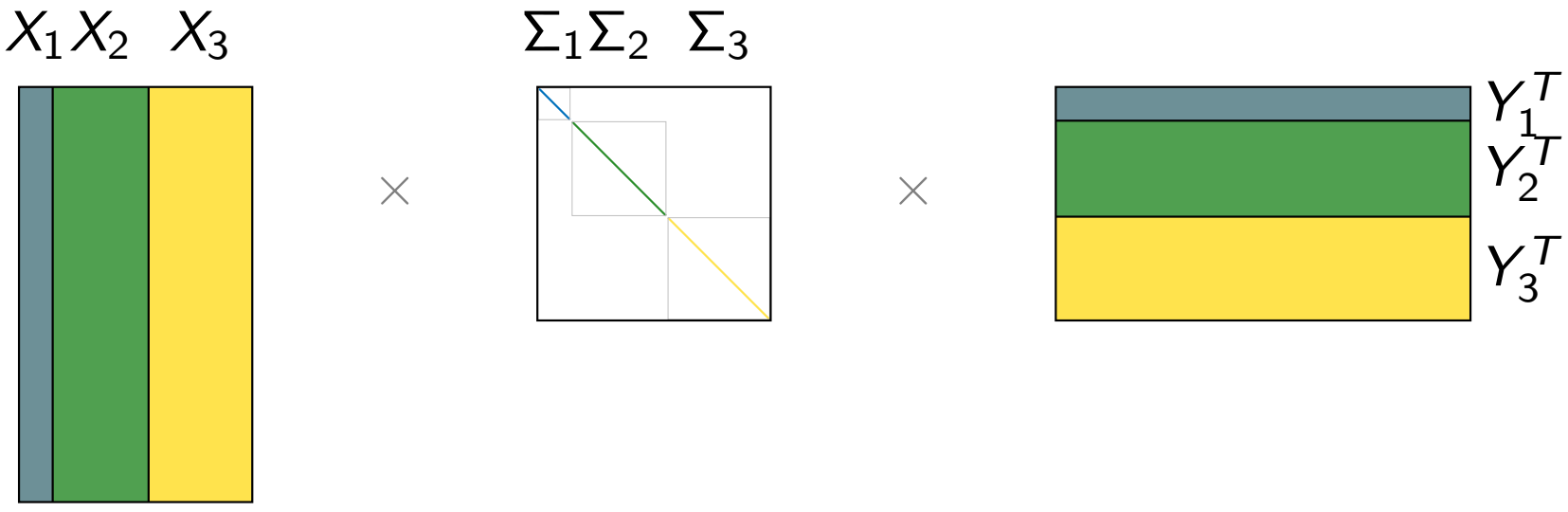
Mixed precision Low-Rank approximation



Truncated SVD with 2 precision formats (fp64, fp32)

- The idea: Converting X_2 and Y_2 to single precision (fp32)
- Criterion for storing columns x_i and y_i in precision fp32: $\sigma_i \leq \frac{\varepsilon}{u_s}\|B\|$
- Error: $\|B - X\Sigma Y\| \lesssim 3\varepsilon\|B\|$

Mixed precision Low-Rank approximation

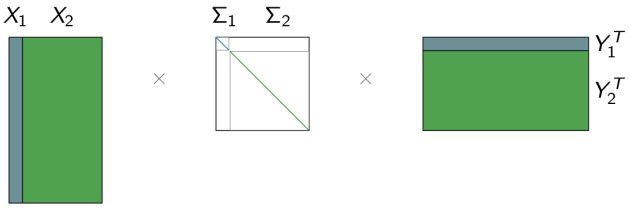


Extension to 3 precision (fp64, fp32, bfloat16)

- Converting X_3 and Y_3 to bfloat16
- Criterion for storing columns x_i and y_i in precision bfloat16: $\sigma_i \leq \frac{\varepsilon}{u_{bf16}}$
- Error: $\|B - X\Sigma Y\| \lesssim 5\varepsilon\|B\|$

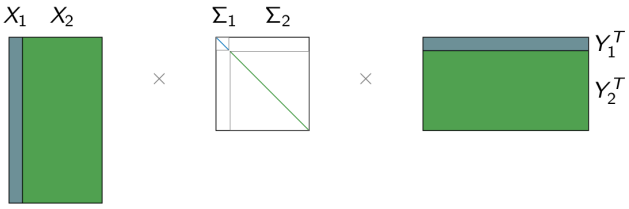
Why does it work? An intuition

- $B = B_1 + B_2$



Why does it work? An intuition

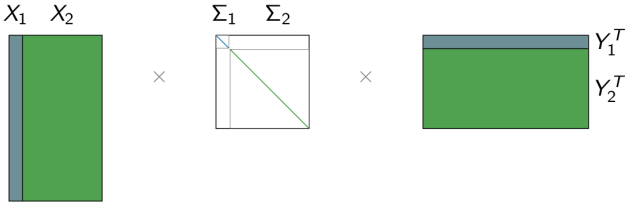
- $B = B_1 + B_2$



- The coefficients of B_2 are small compared to those of B_1

Why does it work? An intuition

- $B = B_1 + B_2$



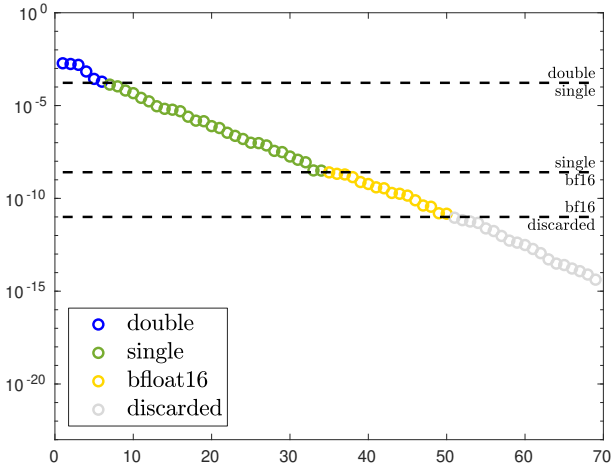
- The coefficients of B_2 are small compared to those of B_1
- Example:

$$\begin{array}{rl} & 1.0\,1\,0\,1\,1\,0\,1 \times 2^{0} \\ + & \qquad 1.1\,0(1\,0\,1\,1\,0) \times 2^{-6} \\ \hline = & 1.0\,1\,1\,0\,0\,0\,0 \end{array}$$

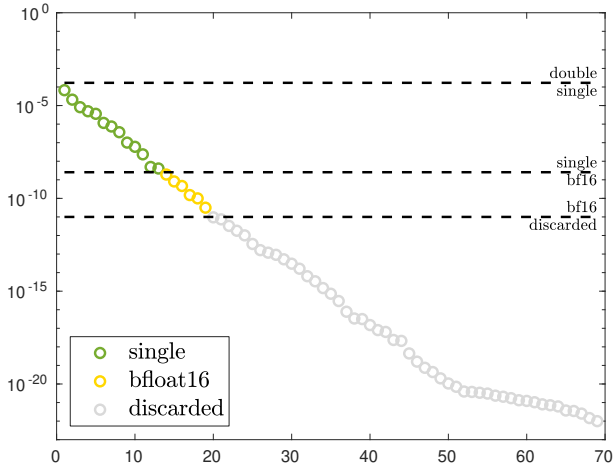
- B_2 can be stored in lower precision, with fewer fraction bits

Distribution of singular values

- A typical example of rapidly decaying singular values for off-diagonal blocks (matrix *perf009*)



`block (12,11)`

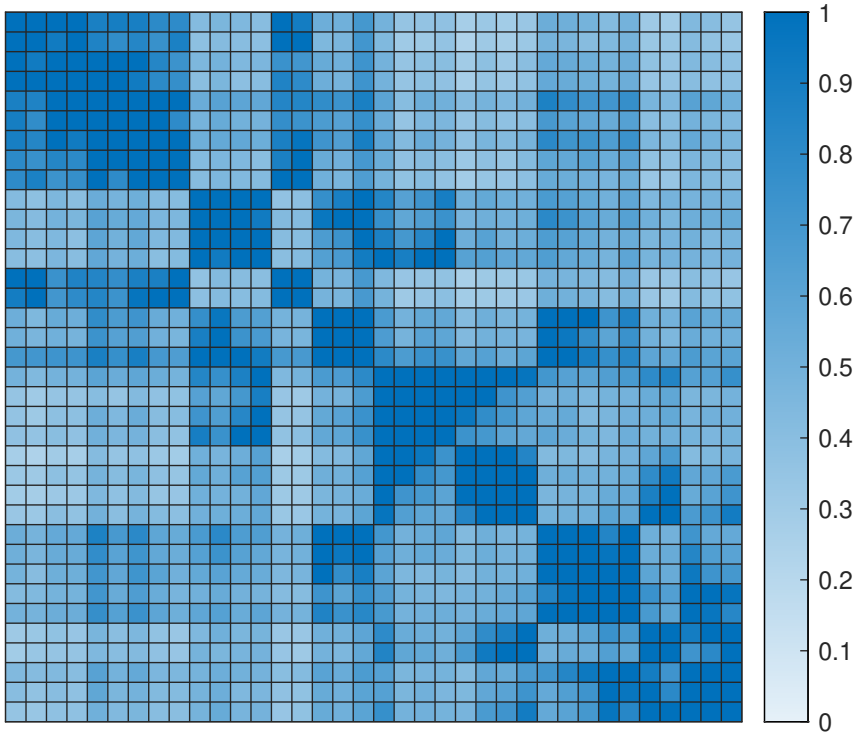


`block (12,25)`

- Most common precision formats:

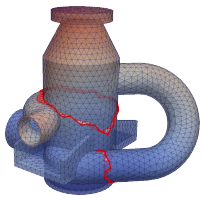
	Signif. bits (t)	Exp.	Range	$u = 2^{-t}$
fp64	53	11	$10^{\pm 308}$	1×10^{-16}
fp32	24	8	$10^{\pm 38}$	6×10^{-8}
fp16	11	5	$10^{\pm 5}$	5×10^{-4}
bfloat16	8	8	$10^{\pm 38}$	4×10^{-3}

Compression gain: an example



Storage cost of the blocks, in percentage of the full-rank blocks

(*perf009*, $\varepsilon = 10^{-10}$)

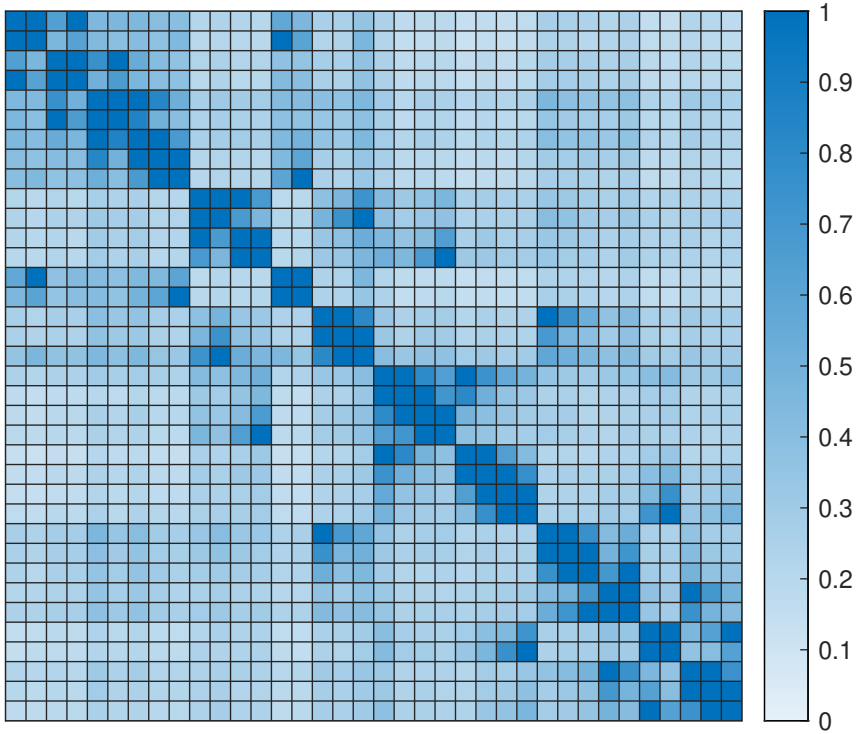


Monoprecision BLR:

- entries in fp64: **100%**

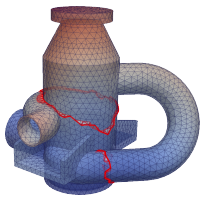
size: 27.9 MBytes

Compression gain: an example



Storage cost of the blocks, in percentage of the full-rank blocks

(*perf009*, $\varepsilon = 10^{-10}$)

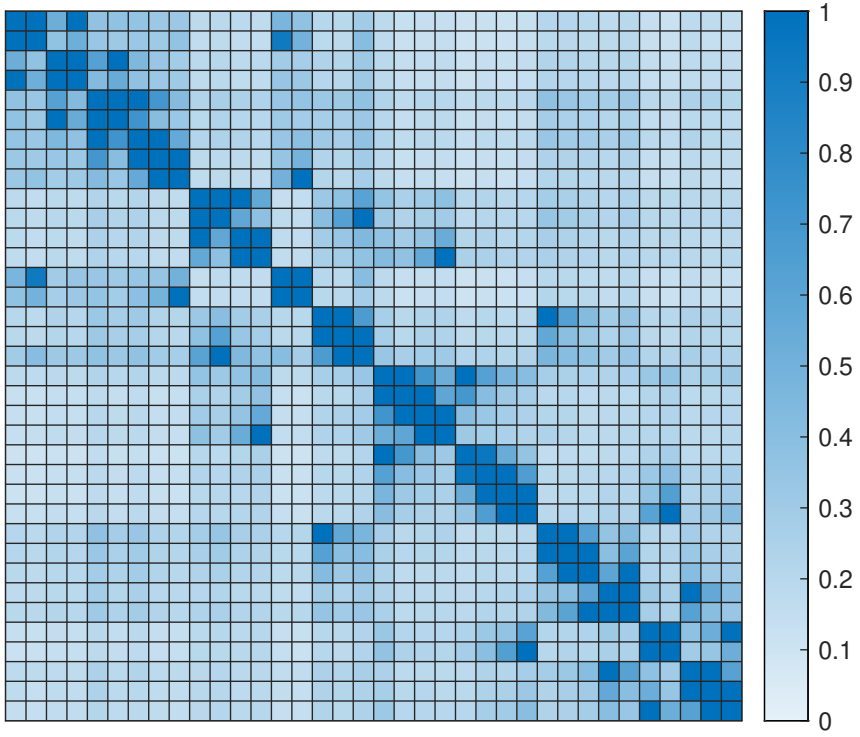


2-precision BLR:

- entries in fp64: **14%**
- entries in fp32: **86%**

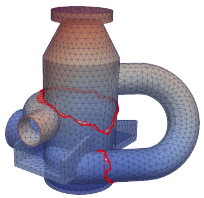
size: 16.4 MBytes
($\times$**1.7** storage gain)

Compression gain: an example



Storage cost of the blocks, in percentage of the full-rank blocks

(*perf009*, $\varepsilon = 10^{-10}$)



3-precision BLR:

- entries in fp64: **13%**
- entries in fp32: **53%**
- entries in bfloat16: **33%**

size: 14.0 MBytes

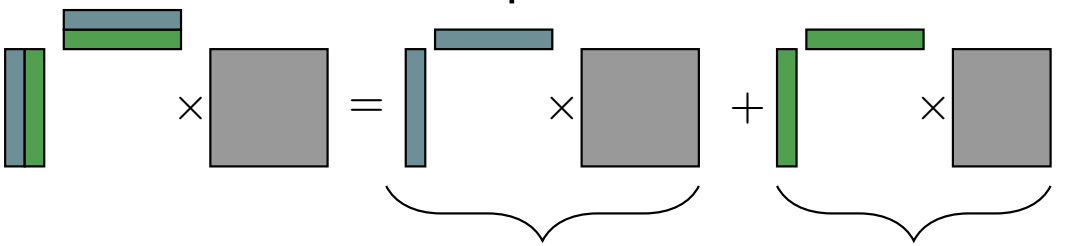
($\times$**2.0** storage gain)

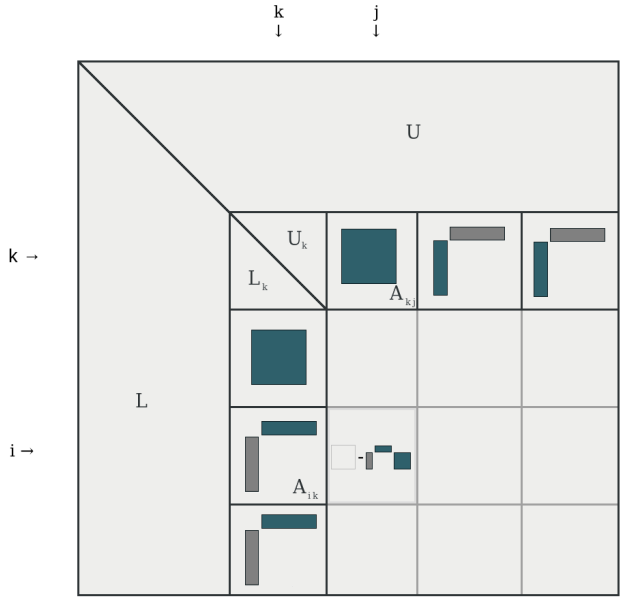
LU factorization algorithm (dense matrix)

- Step k:
  - compute $L_k U_k = A_{kk}$
  - Update formula: for $i, j > k$,
    $A_{ij} \leftarrow A_{ij} - (A_{ik} U_k^{-1}) \times (L_k^{-1} A_{kj})$

- BLR: $A_{ik} \approx X_{ik} Y_{ik}^T$

- Example of kernel in mixed precision:
  LR $\times$ matrix multiplication:





Stability

Traditional LU (Wilkinson)

$$\widehat{L}\widehat{U} = A + \Delta A, \quad \|\Delta A\| \lesssim 3n^3 \rho_n u_1 \|A\|.$$

BLR LU (Higham & Mary)

$$\widehat{L}\widehat{U} = A + \Delta A, \quad \|\Delta A\| \leq \left(c_1 \varepsilon + c_2 \rho_n u_1\right) \|A\|.$$

Mixed precision BLR LU (this work)

$$\widehat{L}\widehat{U} = A + \Delta A, \quad \|\Delta A\| \leq \left(c_1' \varepsilon + c_2' \rho_n u_1\right) \|A\|.$$

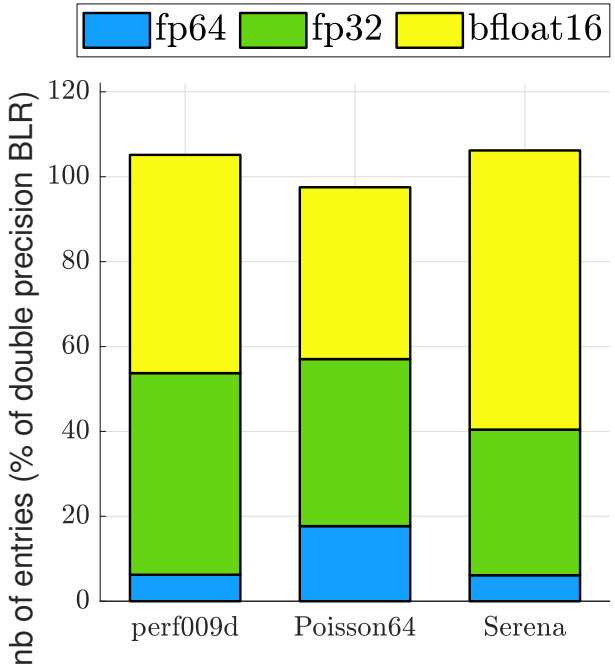
Stability is preserved with mixed precision

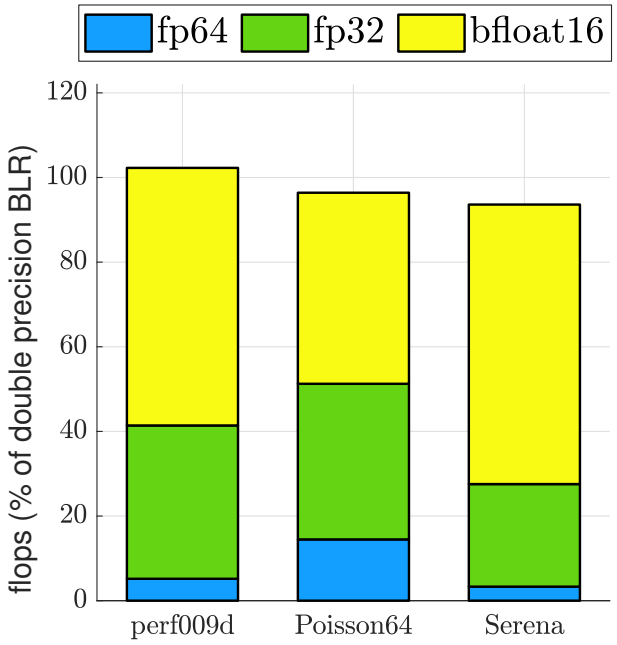
See article: *Mixed Precision Low Rank Approximations and their Application to Block Low Rank LU Factorization*

[5] ρ_n is the growth factor of the LU factorization (often small)

A prototype (for dense matrices)

- Experiments with a Matlab prototype to assess potential gains
- Simulating a LU factorization in 3 precisions: **fp64**, **fp32**, **bfloat16**
- Performance metrics: **storage cost**, **expected time**[6]
- Result: repartition of the precision formats:



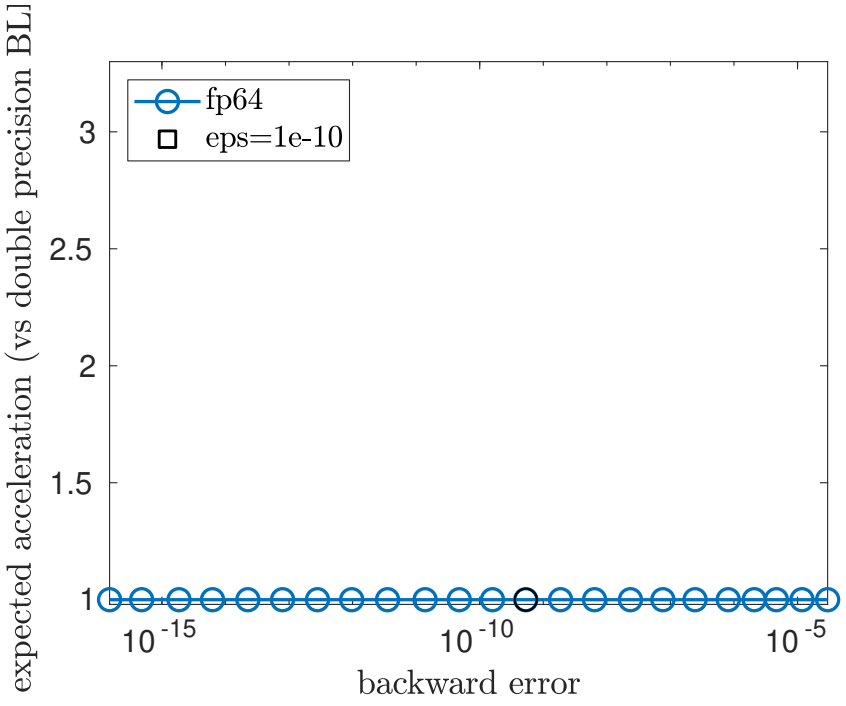


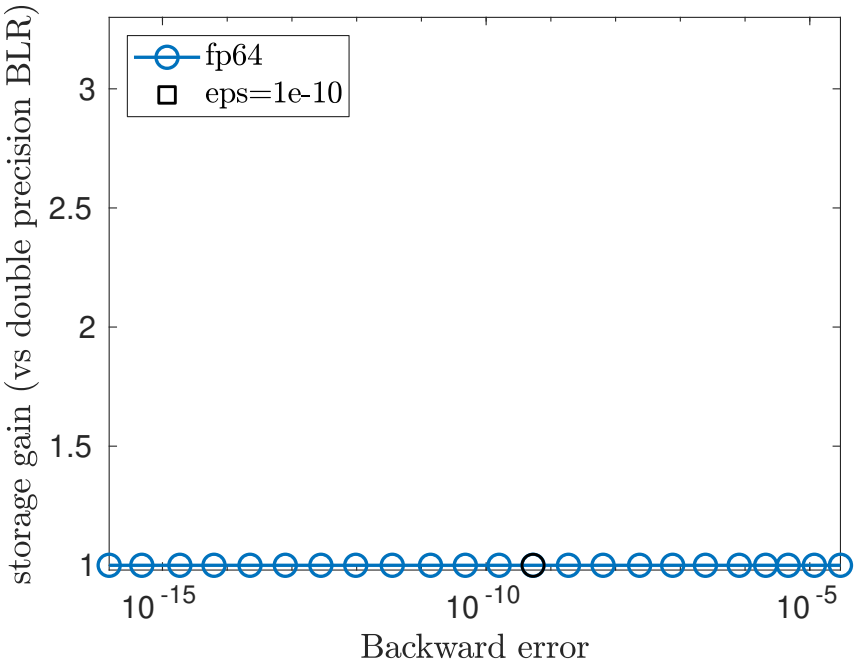
[6] Hypothesis: *1 operation in fp64 = 2 in fp32 = 4 in bfloat16*

[7] Results for $\varepsilon = 10^{-9}$

Tradeoff between performance and accuracy (dense)

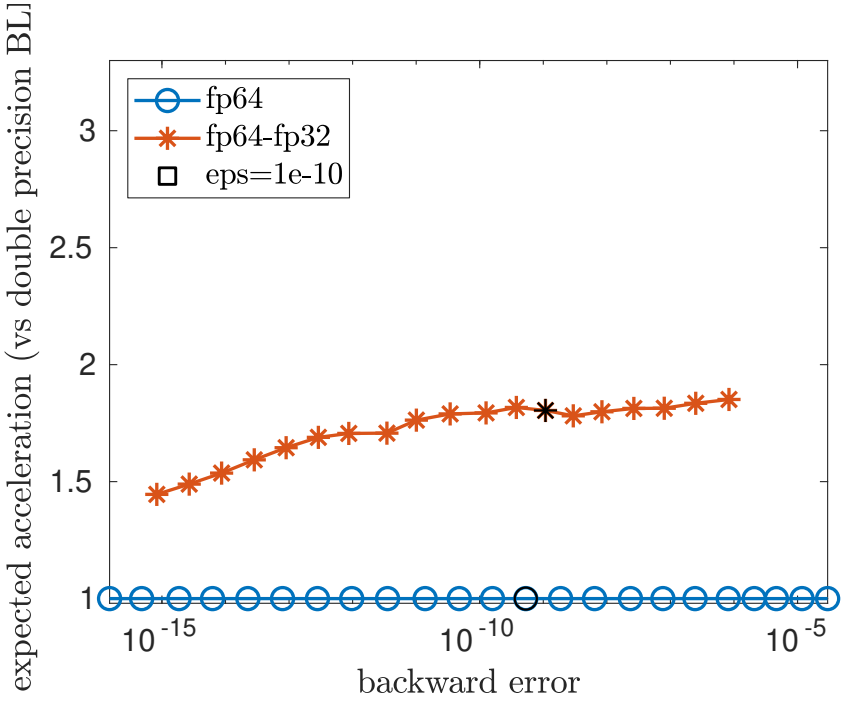
- Better performance
- Loss in accuracy
- Is it still worth it ?
- Example (*perf0009*): performances as a function of the error

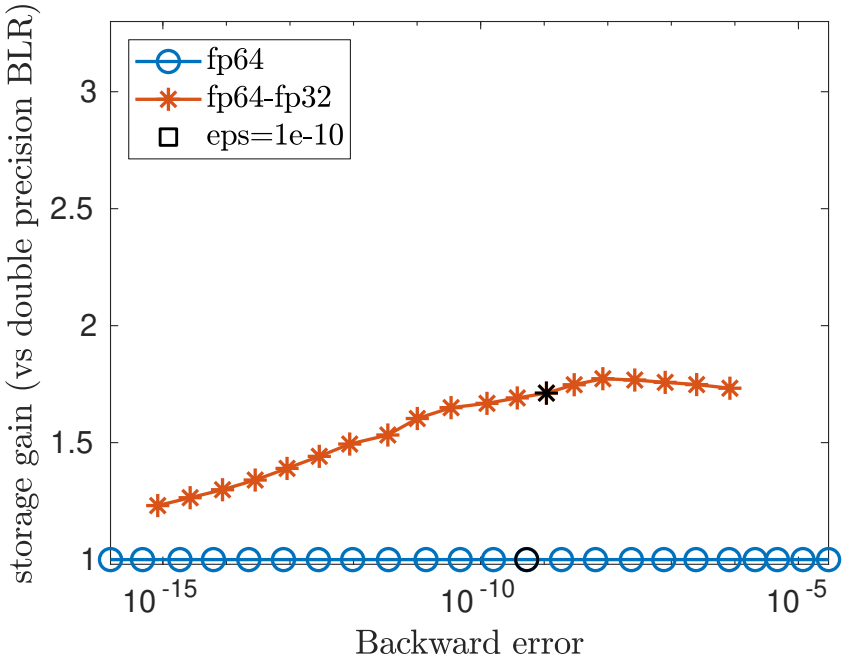




Tradeoff between performance and accuracy (dense)

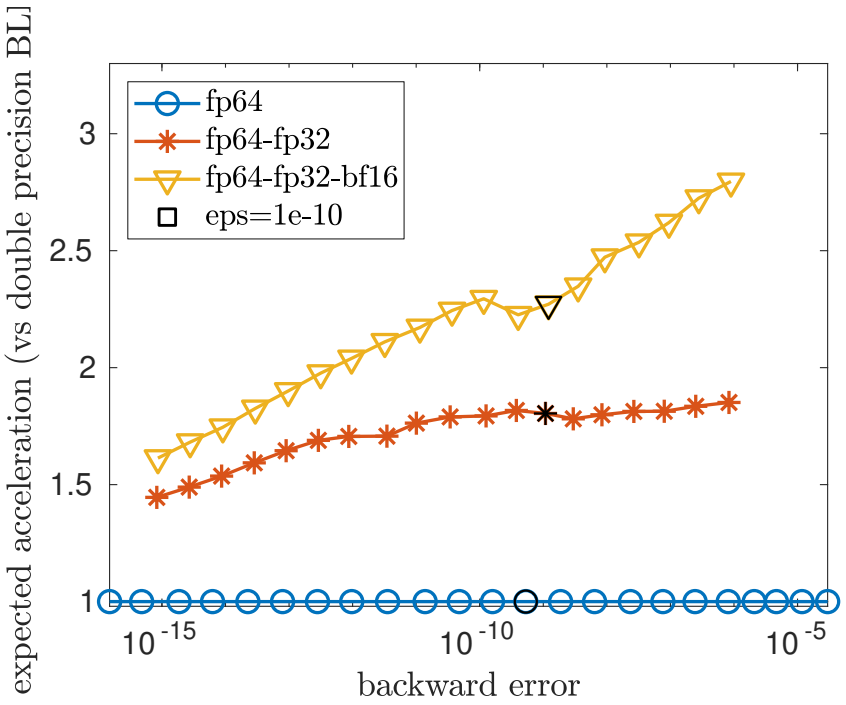
- Better performance
- Loss in accuracy
- Is it still worth it ? Yes
- Example (*perf0009*): performances as a function of the error

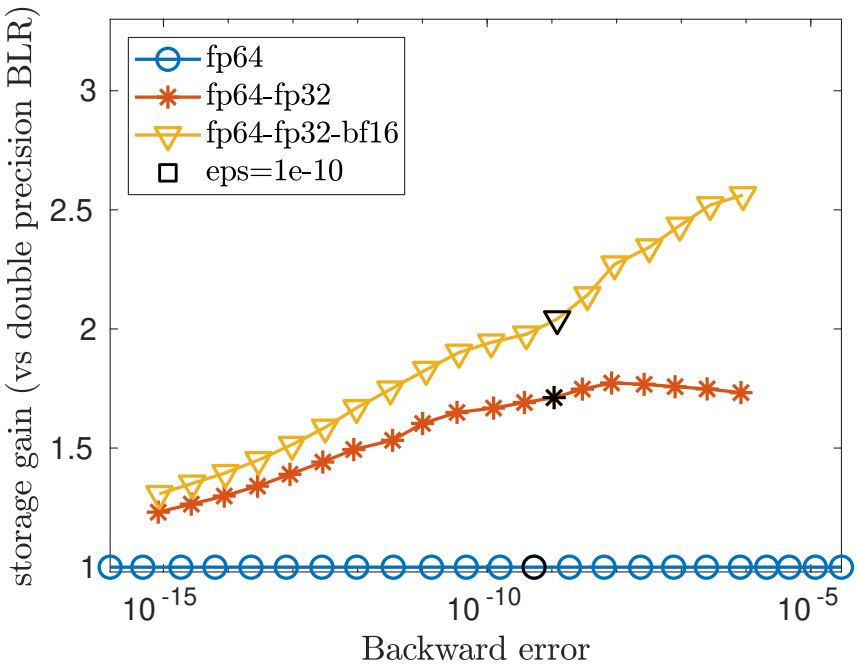




Tradeoff between performance and accuracy (dense)

- Better performance
- Loss in accuracy
- Is it still worth it ? Yes
- Example (*perf0009*): performances as a function of the error



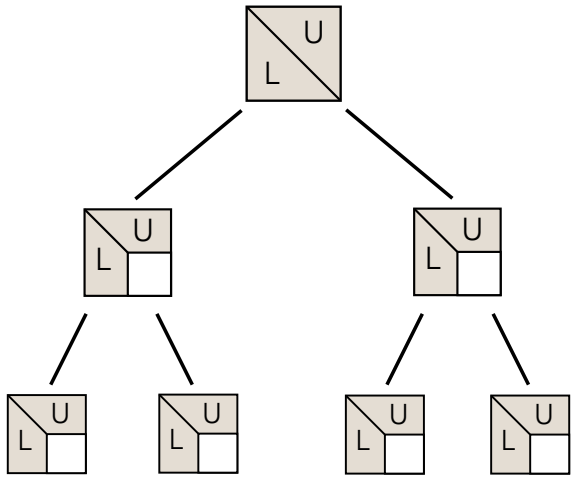


MUMPS: LU factorization

- Multifrontal method:

→ LU factorisation of a large sparse matrix

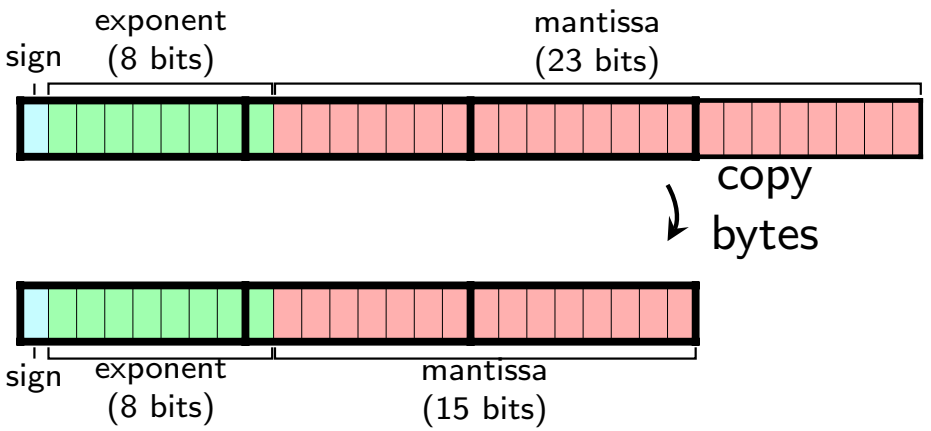
→ partial LU factorization of many small dense matrix

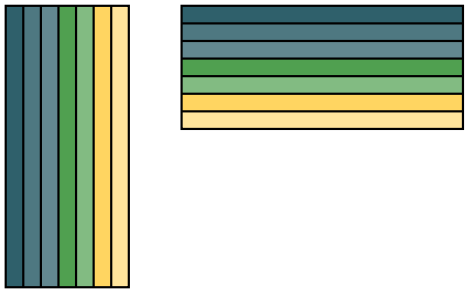


→ Possibility to use BLR compression on those matrices

MUMPS: mixed precision for storage

- Using mixed precision for storage only: the formats do not need to be supported in hardware
- Example: 7 precisions formats, using respectively 16, 24, 32, 40, 48, 56 and 64 bits
- Example: conversion from fp32 to "fp24":





Representation of a low-rank block stored in 7 precisions

MUMPS: storage results

Matrix	precision	Memory peak (GBytes)	Scaled residual
thmgas	fp64	120	6.4E-14
	mixed	86	5.5E-14
perf009	fp64	36	1.3E-10
	mixed	32	1.4E-10
knuckle	fp64	281	1.6E-10
	mixed	236	7.7E-9

$\rightarrow$ Relative gains on memory peak: $\times 1.1$ to $\times 1.4$

Conclusion

- Mixed-precision BLR compression and its use in LU factorization are motivated by an **error analysis**
- An **article** submitted to IMA Journal of Numerical Analysis: *Mixed Precision Low Rank Approximations and their Application to Block Low Rank LU Factorization* 
- A first implementation in sparse solver MUMPS, achieving a **storage reduction** up to 40%
- Some future works:
  - We could consider **fp16** instead of bfloat16 ($\Rightarrow$ **scaling**)
  - Continue the implementation in MUMPS, aiming for **time gains** in the factorization

Appendix: Low precision arithmetics

	Signif. bits(t)	Exponent bits	Range	$u = 2^{-t}$
fp64	53	11	$10^{\pm 308}$	1×10^{-16}
fp32	24	8	$10^{\pm 38}$	6×10^{-8}
fp16	11	5	$10^{\pm 5}$	5×10^{-4}
bfloat16	8	8	$10^{\pm 38}$	4×10^{-3}

Half precision increasingly supported by hardware:

- Fp16 used by NVIDIA GPUs, AMD Radeon Instinct MI25 GPU, ARM NEON, Fujitsu A64FX ARM
- Bfloat16 used by Google TPU, NVIDIA GPUs, Arm, Intel

Great benefits:

- Reduced storage, data movement, and communications
- Increased speed , e.g., with GPU Tensor Cores:
  fp32 $\rightarrow$ fp16 speedup evolution:
  P100: 2× V100: 8× A100: 16× H100: 16×
- Reduced **energy** consumption (5× with fp16, 9× with bfloat16!)

$\rightarrow$ Motivations to use mixed-precision algorithms

Appendix: Test matrices for Matlab prototype

- Dense matrices obtained from the root separator (Schur complement) of sparse matrices
- $\varepsilon = 10^{-9}$

Matrix	Application	N	block size
P64	Poisson equation	4k	128
perf009	Elastic computation of a RIS pump under internal pressure (EDF, code_aster)	2k	64
Serena	Gas resevoir simulation for CO2 sequestration	16k	256

Appendix: Matrices for MUMPS

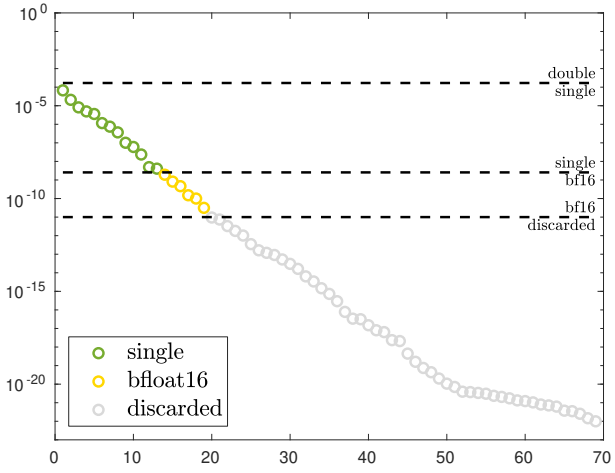
Matrix	N	NNZ	SYM[8]
thmgas	4.9M	471M	0
perf009	5.4M	209M	2
knuckle	8.5M	363M	2

- *thmgas*: taking gas into account in storage of nuclear waste (code_aster, thermo-hydro-mechanical couplings)
- *knuckle*: a matrix from Altair OptiStruct (structural mechanics)
- *perf009*: a RIS pump under internal pressure: elastic computation (EDF, code_aster)

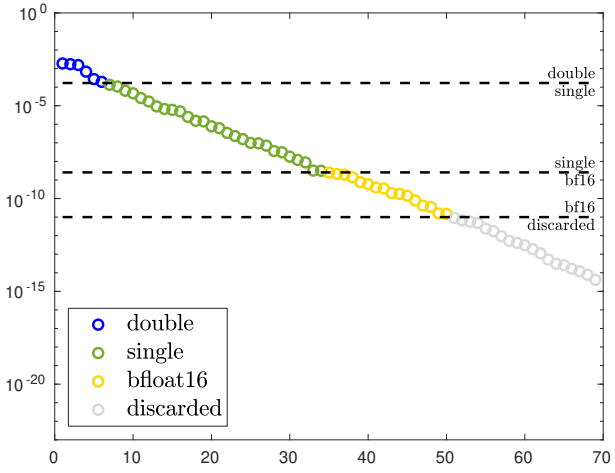
[8] SYM = **0: unsymmetric**; 1: SDP; **2: symmetric**

Appendix: Distribution of singular values

- A typical example of rapidly decaying singular values for off-diagonal blocks (matrix *perf009*)



block (12,25)

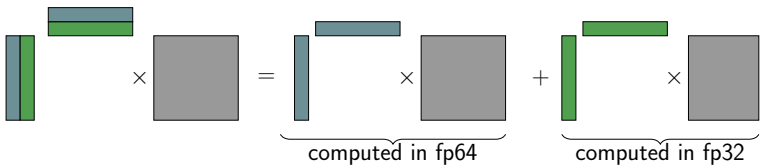


block (12,11)

- Classic criterion for low-rank admissibility : $r < n/2$ (i.e. storage reduction)
- A criterion for mixed-precision low-rank admissibility : $r_d + 0.5r_s + 0.25r_h < n/2$ (i.e. storage reduction)

Appendix: mixed precision for computation in MUMPS

- Triangular solve step: $LX = B$, whereL is BLR
- Accelerated using 2 precisions for computations (in LR $\times$ matrix product):



- Some early results[9]:

precision	Avg time in solve forward (s)	Scaled residual
double	0.76	2.9E-12
mixed	0.67	3.9E-12
single	0.46	5.2E-8

[9] on matrix Queen_4147 from SuiteSparse, for $\varepsilon = 10^{-9}$)