Mixed-precision in iterative refinement and low-rank approximations for the solution of linear systems

P. Amestoy, O. Boiteau, *A. Buttari*, M. Gerest, F. Jézéquel, J.-Y. L'Excellent, N. J. Higham, T. Mary, B. Vieublé

ISC 2021, Numerical Algorithms and Libraries for Exascale

Widening range of arithmetics

	ID	Signif. bits	Exp. bits	Range	Unit roundoff u
fp128	Q	113	15	$10^{\pm 4932}$	1×10^{-34}
fp64	D	53	11	$10^{\pm 308}$	1×10^{-16}
fp32	S	24	8	$10^{\pm 38}$	6×10^{-8}
fp16	H	11	5	$10^{\pm 5}$	5×10^{-4}
bfloat16	B	8	8	$10^{\pm 38}$	4×10^{-3}

- Half precision increasingly **supported by hardware** (NVIDIA, Arm, AMD, Intel soon, ...)
- **Great benefits** :
  - Reduced storage, data movement, and communications
  - Faster computations : fp32 $\rightarrow$ fp16 speedup evolution with **GPU Tensor Cores** : P100 : **2**× V100 : **8**× A100 : **16**×
  - Reduced energy consumption
- However, low precision = low accuracy
- Motivation for **mixed precision algorithms**.

Layout

Part 1 Five-precision iterative refinement

B. Vieuble's PhD (IRIT, INPT)

Amestoy, Buttari, Higham, L'Excellent, Mary, Vieublé. *"Five-Precision GMRES-based iterative refinement"*. In : Preprint available on HAL (ID : hal-03190686).

Part 2 Mixed precision block low-rank

M. Gerest's PhD (EDF, LIP6)

Amestoy, Boiteau, Buttari, Gerest, Jézéquel, L'Excellent, Mary. *"Mixed Precision Low Rank Approximations and their Application to Block Low Rank LU Factorization"*. In : Preprint available on HAL (ID : hal-03251738).

Five-precision iterative refinement

Generalized iterative refinement

Algorithm Generalized iterative refinement

1 :	Compute the LU factorization $A = LU$	(u_f)
2 :	Solve $Ax_0 = b$	(u_f)
3 :	**while not** converged **do**	
4 :	Compute $r_i = b - Ax_i$	(u_r)
5 :	Solve $Ad_i = r_i$	(u_s)
6 :	Compute $x_{i+1} = x_i + d_i$	(u)
7 :	**end while**	

- The solver at step 5 is arbitrary.
- u_s expresses the precision of the computed solution d_i provided by this solver ($\neq$ unit roundoff of an arithmetic precision).

E. Carson and N. J. Higham. *"Accelerating the solution of linear systems by iterative refinement in three precisions"*. In : SIAM SISC, 2018.

Generalized iterative refinement

Algorithm Generalized iterative refinement

1 : Compute the LU factorization $A = LU$ (u_f)
2 : Solve $Ax_0 = b$ (u_f)
3 : **while not** converged **do**
4 : Compute $r_i = b - Ax_i$ (u_r)
5 : Solve $Ad_i = r_i$ (u_s)
6 : Compute $x_{i+1} = x_i + d_i$ (u)
7 : **end while**

Two main properties determined by the set of precisions :

- **The convergence condition :** the maximal value of $\kappa(A)$ for which convergence is guaranteed. (u_f, u_s)
- **The limiting accuracies :** the accuracies at which the forward and backward errors converge. (u, u_r)

LU-IR3

Algorithm LU-based iterative refinement in three precisions

1 : Compute the LU factorization $A = LU$ (u_f)
2 : Solve $Ax_0 = b$ (u_f)
3 : **while not** converged **do**
4 : Compute $r_i = b - Ax_i$ (u_r)
5 : Solve $Ad_i = r_i$ by $d_i = \hat{U}^{-1}\hat{L}^{-1}r_i$. (u_f)
6 : Compute $x_{i+1} = x_i + d_i$ (u)
7 : **end while**

Step 5 : Solver LU – $u_s \equiv u_f$.

E. Carson and N. J. Higham. *"Accelerating the solution of linear systems by iterative refinement in three precisions"*. In : SIAM SISC, 2018.

LU-IR3

Algorithm LU-based iterative refinement in three precisions

1 : Compute the LU factorization $A = LU$ (u_f)
2 : Solve $Ax_0 = b$ (u_f)
3 : **while not** converged **do**
4 : Compute $r_i = b - Ax_i$ (u_r)
5 : Solve $Ad_i = r_i$ by $d_i = \hat{U}^{-1}\hat{L}^{-1}r_i$. (u_f)
6 : Compute $x_{i+1} = x_i + d_i$ (u)
7 : **end while**

	Convergence condition	Forward error
LU-IR3	$\kappa(A) < u_f^{-1}$	$u_r\kappa(A) + u$

Very low precision factorization (e.g fp16, bfloat16) leads to a very restrictive convergence condition for LU-IR3 (e.g 2×10^3).

GMRES-IR3

Algorithm GMRES-based iterative refinement in three precisions

1 : Compute the LU factorization $A = LU$ (u_f)

2 : Solve $Ax_0 = b$ (u_f)

3 : **while not** converged **do**

4 : Compute $r_i = b - Ax_i$ (u_r)

5 : Solve $\widetilde{A}d_i = \hat{U}^{-1}\hat{L}^{-1}Ad_i = \hat{U}^{-1}\hat{L}^{-1}r_i$ by GMRES at precision (u) with matrix vector products with $\widetilde{A}$ at precision (u^2).

6 : Compute $x_{i+1} = x_i + d_i$ (u)

7 : **end while**

Step 5 : Preconditioned GMRES in two precision – $u_s \equiv u$.

E. Carson and N. J. Higham. *"A new analysis of iterative refinement and its application to accurate solution of ill-conditioned sparse linear systems"*. In : SIAM SISC, 2017.

GMRES-IR3

Algorithm GMRES-based iterative refinement in three precisions

1 : Compute the LU factorization $A = LU$ (u_f)
2 : Solve $Ax_0 = b$ (u_f)
3 : **while not** converged **do**
4 : Compute $r_i = b - Ax_i$ (u_r)
5 : Solve $\widetilde{A}d_i = \hat{U}^{-1}\hat{L}^{-1}Ad_i = \hat{U}^{-1}\hat{L}^{-1}r_i$ by GMRES at precision (u) with matrix vector products with $\widetilde{A}$ at precision (u^2).
6 : Compute $x_{i+1} = x_i + d_i$ (u)
7 : **end while**

	Convergence condition	Forward error
LU-IR3	$\kappa(A) < u_f^{-1}$	$u_r\kappa(A) + u$
GMRES-IR3	$\kappa(A) < u^{-1/2}u_f^{-1}$	$u_r\kappa(A) + u$

If u_f is fp16, then the condition on LU-IR3 is $\mathbf{2 \times 10^3}$, on GMRES-IR3 is $\mathbf{2 \times 10^{11}}$!

Practical issues of GMRES-IR3

In GMRES-IR3, LU solves are performed at precision u^2 : this is a **major practical issue**.

Practical issues of GMRES-IR3

In GMRES-IR3, LU solves are performed at precision u^2 : this is a **major practical issue**.

- Higher cost per iteration.
- If $u = \text{fp64} \Rightarrow u^2 = \text{fp128}$: not hardware support nor efficient BLAS libraries for quad precision.
- Potentially cast the LU factors from precision u_f to u^2
  $\Rightarrow$ large memory increase

Other issue : Do we need to run the other GMRES operations in precision u ?

Practical issues of GMRES-IR3

In GMRES-IR3, LU solves are performed at precision u^2 : this is a **major practical issue**.

- Higher cost per iteration.
- If $u = \text{fp64} \Rightarrow u^2 = \text{fp128}$: not hardware support nor efficient BLAS libraries for quad precision.
- Potentially cast the LU factors from precision u_f to u^2
  $\Rightarrow$ large memory increase

Other issue : Do we need to run the other GMRES operations in precision u ?

$\Rightarrow$ What if we **relax the precision $\mathbf{u^2}$** on the preconditioning **and u** on the rest of the operations ?

GMRES-IR5

Algorithm GMRES-IR3

1 : Compute the LU factorization $A = LU$ (u_f)

2 : Solve $Ax_0 = b$ (u_f)

3 : **while not** converged **do**

4 : Compute $r_i = b - Ax_i$ (u_r)

5 : Solve $\widetilde{A}d_i = \hat{U}^{-1}\hat{L}^{-1}Ad_i = \hat{U}^{-1}\hat{L}^{-1}r_i$ by GMRES at precision $(u\ )$ with matrix vector products with $\widetilde{A}$ at precision (u^2).

6 : Compute $x_{i+1} = x_i + d_i$ (u)

7 : **end while**

GMRES-IR5

Algorithm GMRES-IR3

1 : Compute the LU factorization $A = LU$ (u_f)
2 : Solve $Ax_0 = b$ (u_f)
3 : **while not** converged **do**
4 : Compute $r_i = b - Ax_i$ (u_r)
5 : Solve $\widetilde{A}d_i = \hat{U}^{-1}\hat{L}^{-1}Ad_i = \hat{U}^{-1}\hat{L}^{-1}r_i$ by GMRES at precision $(u \quad)$ with matrix vector products with $\widetilde{A}$ at precision (u^2).
6 : Compute $x_{i+1} = x_i + d_i$ (u)
7 : **end while**

GMRES-IR5

Algorithm GMRES-IR5

1 : Compute the LU factorization $A = LU$ (u_f)
2 : Solve $Ax_0 = b$ (u_f)
3 : **while not** converged **do**
4 : Compute $r_i = b - Ax_i$ (u_r)
5 : Solve $\widetilde{A}d_i = \hat{U}^{-1}\hat{L}^{-1}Ad_i = \hat{U}^{-1}\hat{L}^{-1}r_i$ by GMRES at precision $(u_{\mathbf{g}})$ with matrix vector products with $\widetilde{A}$ at precision $(u_{\mathbf{p}})$.
6 : Compute $x_{i+1} = x_i + d_i$ (u)
7 : **end while**

- $u_{\mathbf{p}}$: precision at which we apply the **preconditioned** matrix-vector products.
- $u_{\mathbf{g}}$: precision at which we apply the other **GMRES** operations.

Possibly $u_p > u^2$ (and $u_g > u$).

Preconditioned MGS-GMRES in 2 precisions

Theorem (Stability of preconditioned MGS-GMRES in 2 precisions)

Consider solving a preconditioned linear system

$$\widetilde{A}d = s, \quad \widetilde{A} = \hat{U}^{-1}\hat{L}^{-1}A, \quad A \in \mathbb{R}^{n\times n},$$

with a MGS-GMRES in precision u_g except for the products with $\widetilde{A}$ applied in precision u_p.

The computed solution $\hat{d}$ achieves a backward error of order

$$\mathbf{u_s \equiv u_g + u_p \kappa(A)}$$

$\Rightarrow$ It generalizes the **backward stability** of **MGS-GMRES** to a preconditioned MGS-GMRES in **2 precisions**.

C. Paige, M. Rozložník and Z. Strakoš. *"Modified Gram-Schmidt (MGS), least squares, and backward stability of MGS-GMRES"*. In : SIAM SIMAX, 2006.

Convergence condition of GMRES-IR5

IR	Convergence condition
LU-IR3	$\kappa(A)u_f \ll 1$
GMRES-IR5	$(\mathbf{u_g} + \mathbf{u_p}\kappa(\mathbf{A}))\kappa(\mathbf{A})^2\mathbf{u_f^2} \ll \mathbf{1}$
GMRES-IR3	$\kappa(A)u^{1/2}u_f \ll 1$

If u_f is fp16, the condition on LU-IR3 is $\mathbf{2}\times\mathbf{10^3}$, on GMRES-IR5 (with $u_g = u_p =$ fp64) is 3×10^7, on GMRES-IR3 is 2×10^{11}

Meaningful combinations

With five arithmetics (fp16, bfloat16, fp32, fp64, fp128) GMRES-IR5 can be formulated in over **3000 different variants !**

They are not all relevant !

Filter principle : Useless to have high precision when we can use low precision without impacting the limiting accuracy and convergence condition.

Filtering rules

- $u^2 \leq u_r \leq u \leq u_f$
- $u_p \leq u_g$
- $u_p < u_f$
- $u_p < u$, $u_p = u$, and $u_p > u$
- $u_g = u$ and $u_g > u$
- $u_g < u_f$, $u_g = u_f$, and $u_g > u_f$

These rules are based on the limiting accuracy and convergence condition formulas.

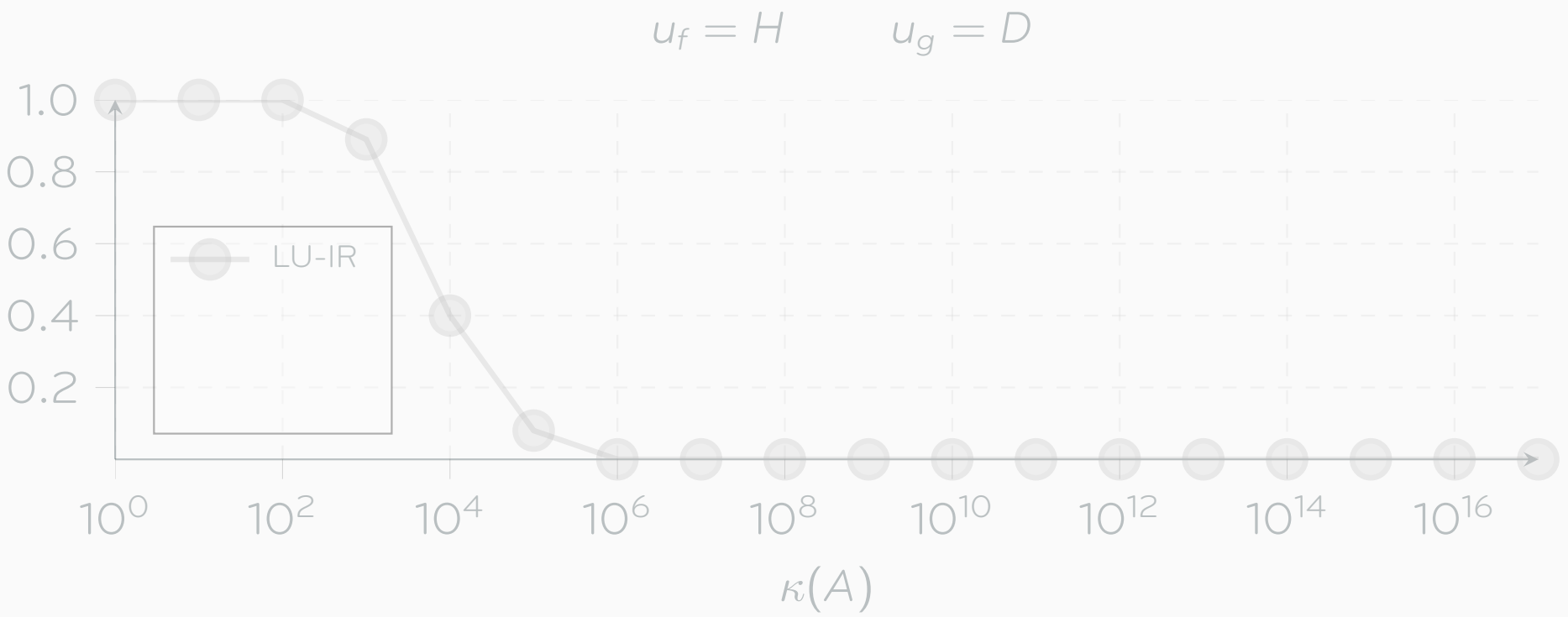
Theoretical robustness over $\kappa(A)$

u_g	u_p	Convergence Condition $\max(\kappa(A))$
LU-IR3		2×10^3
B	S	3×10^4
H	S	4×10^4
H	D	9×10^4
S	D	8×10^6
D	D	3×10^7
GMRES-IR3		2×10^{11}

Meaningful combinations of GMRES-IR5 for $u_f = H$ and $u = D$.

Five combinations between LU-IR3 and GMRES-IR3 $\Rightarrow$ More **flexible** precisions choice to fit at best the **hardware constraints** and the **problem difficulty**.

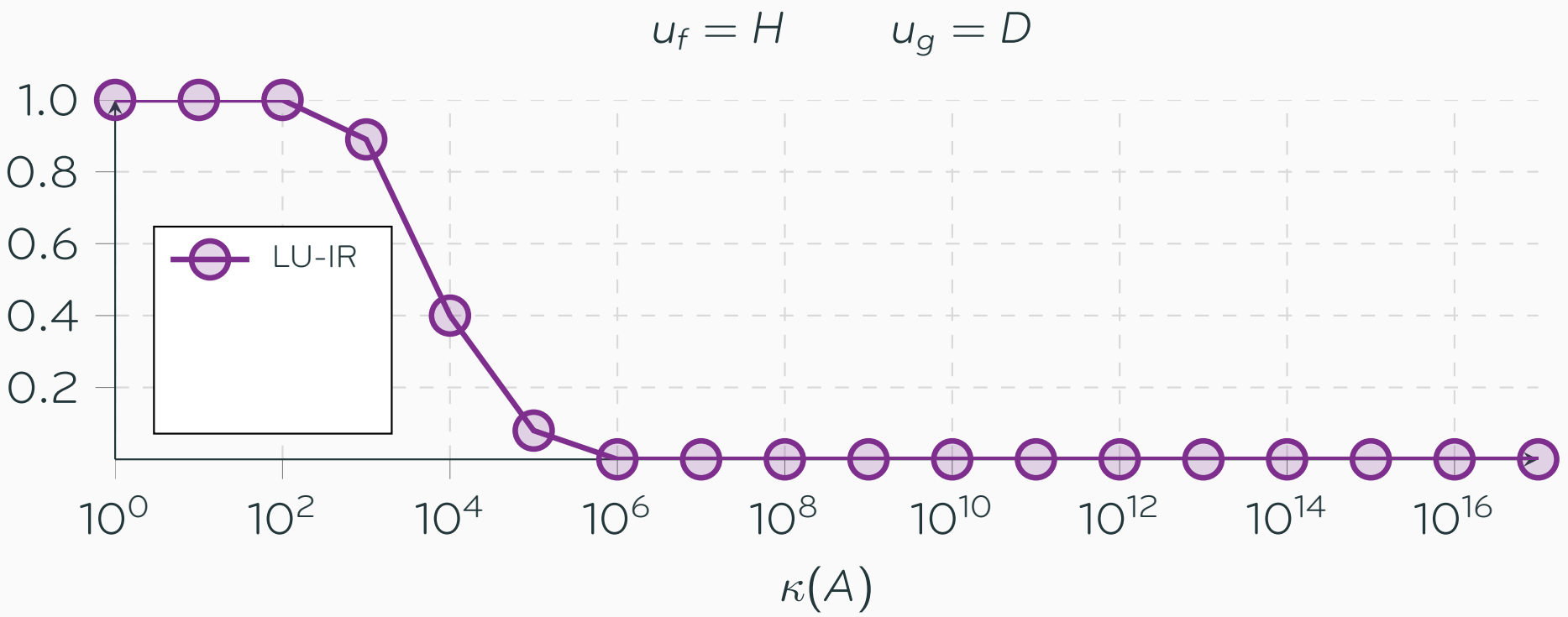
Experiments : robustness over $\kappa(A)$



Robustness w.r.t $\kappa(A)$:

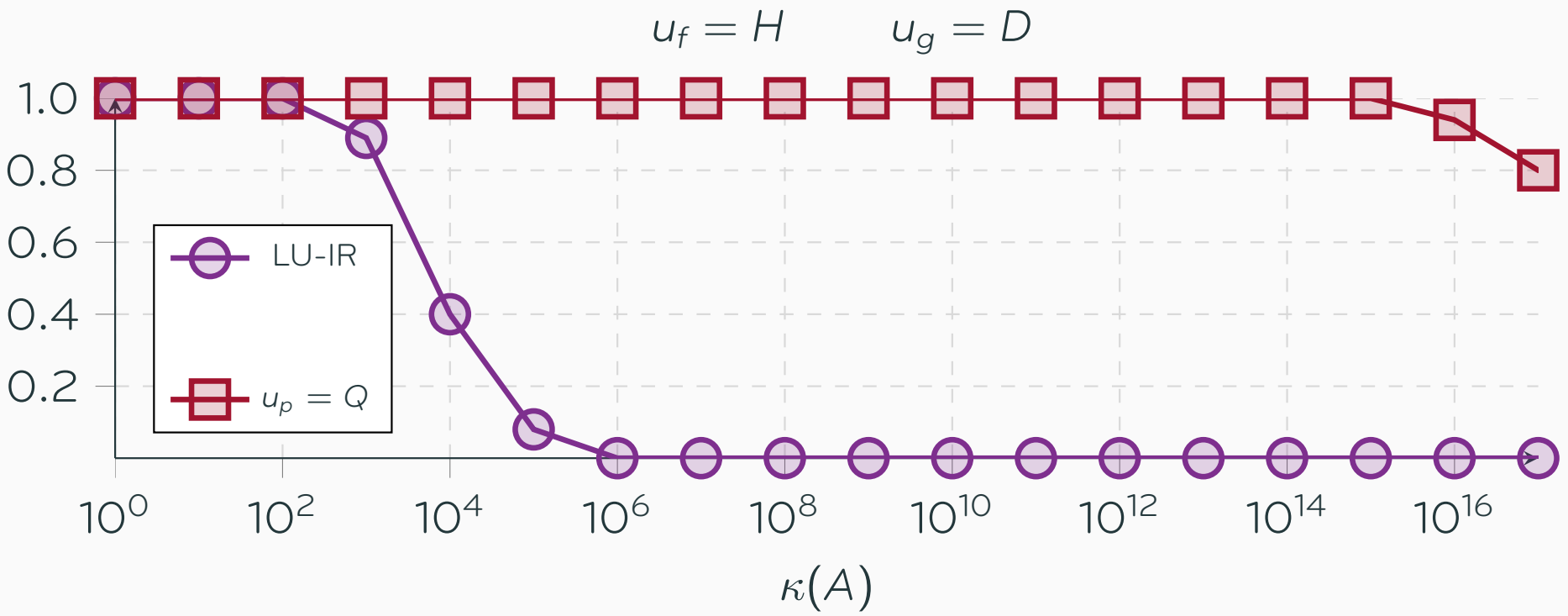
- $\mathbf{u} = D$, $\mathbf{u_r} = Q$, and $\mathbf{u_f} = H$ fixed, $\mathbf{u_g}$ and $\mathbf{u_p}$ in GMRES varying
- iterations are stopped when fwd error converges to limiting accuracy (i.e., 10^{-16})
- measure success rate on 100, 50×50 randsvd dense matrices

Experiments : robustness over $\kappa(A)$



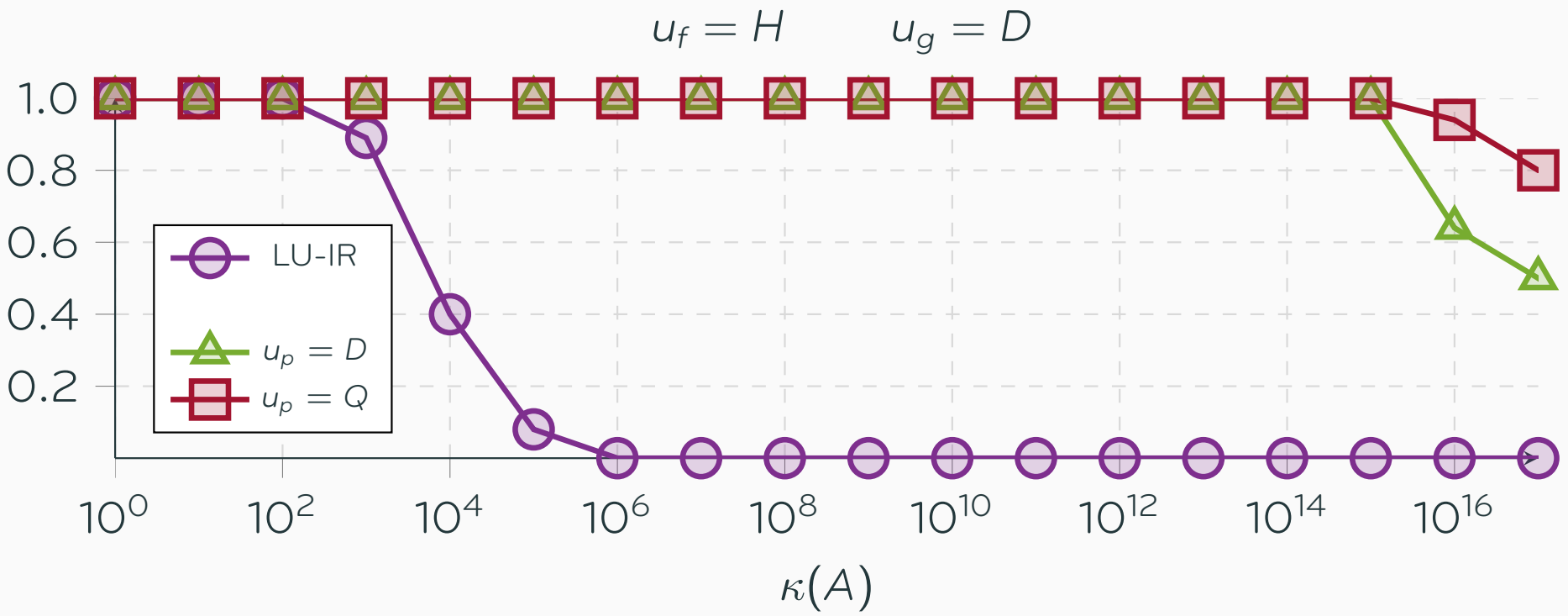
- LU-IR3 can only handle relatively well conditioned problems, as expected ($\kappa(A) \leq 2 \times 10^3$)

Experiments : robustness over $\kappa(A)$



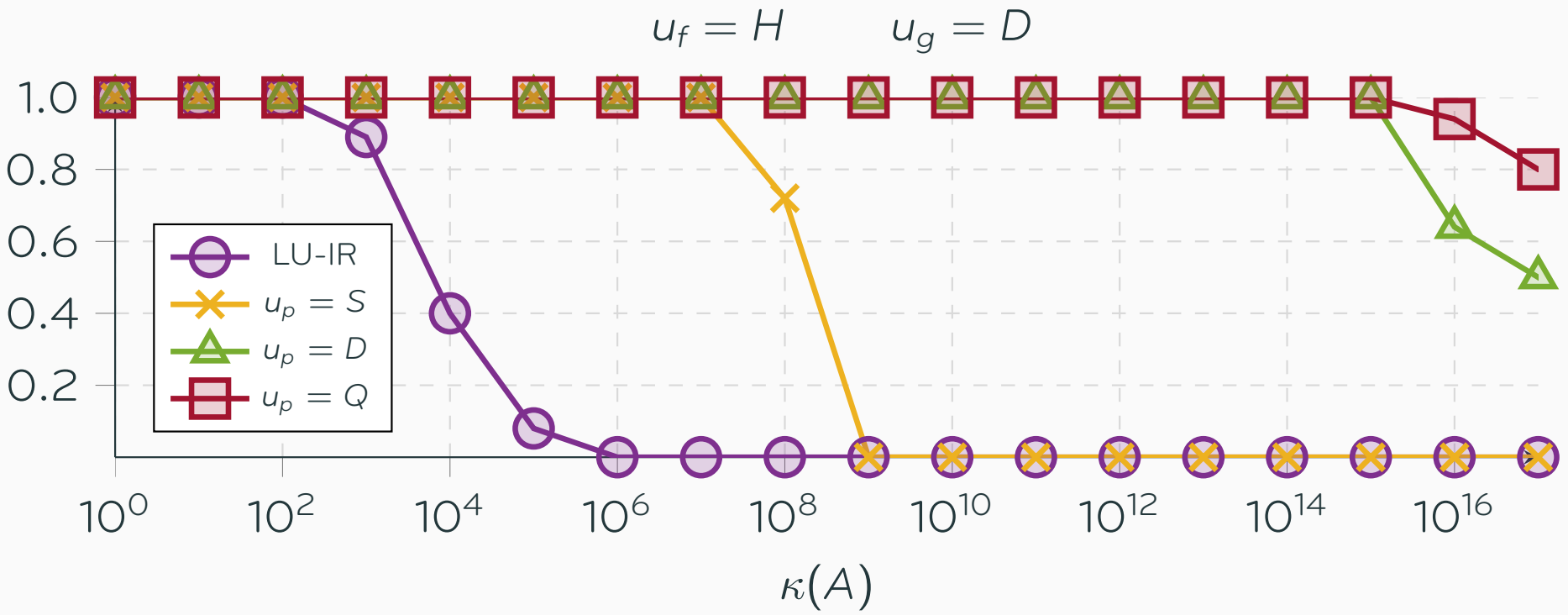
- GMRES-IR5 with $u_g = D$
  - if $u_p = Q$ (i.e., GMRES-IR3) 100% success up to $\kappa(A) \simeq 10^{16}$

Experiments : robustness over $\kappa(A)$



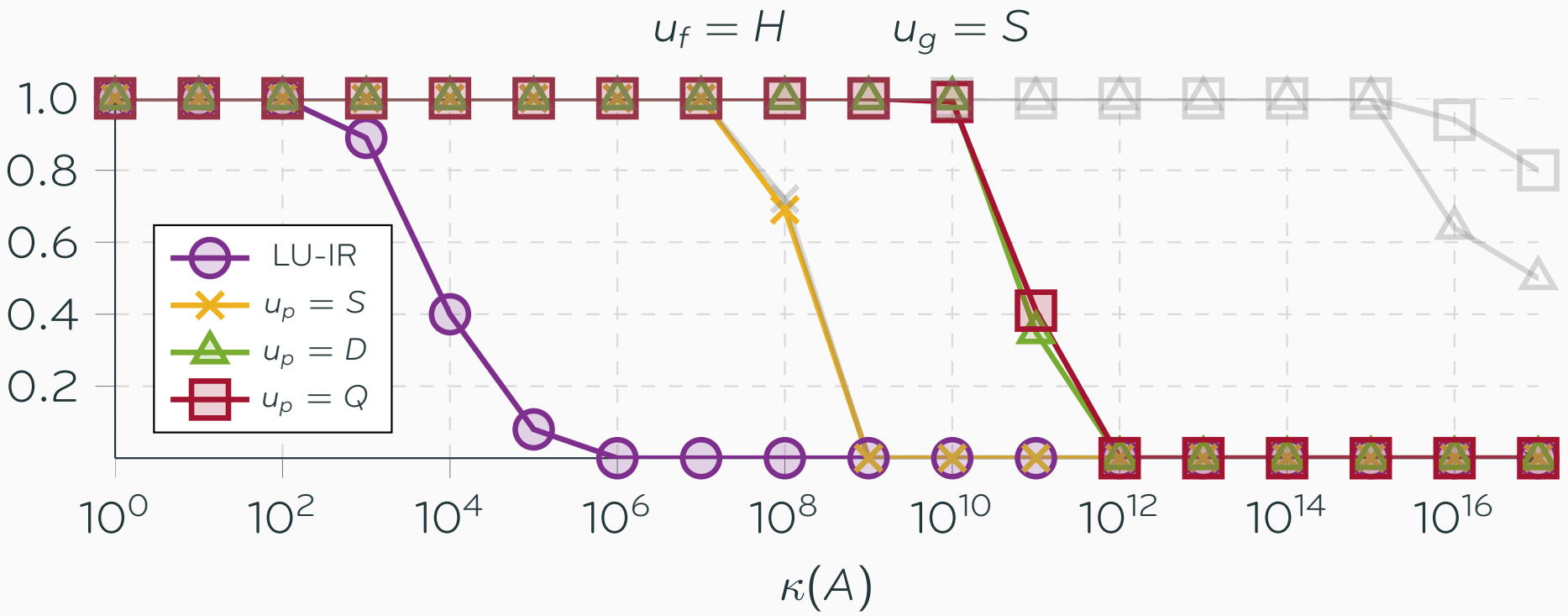
- GMRES-IR5 with $u_g = D$
  - if $u_p = Q$ (i.e., GMRES-IR3) 100% success up to $\kappa(A) \simeq 10^{16}$
  - if $u_p = D$ only little robustness loss but (potentially) much higher efficiency

Experiments : robustness over $\kappa(A)$



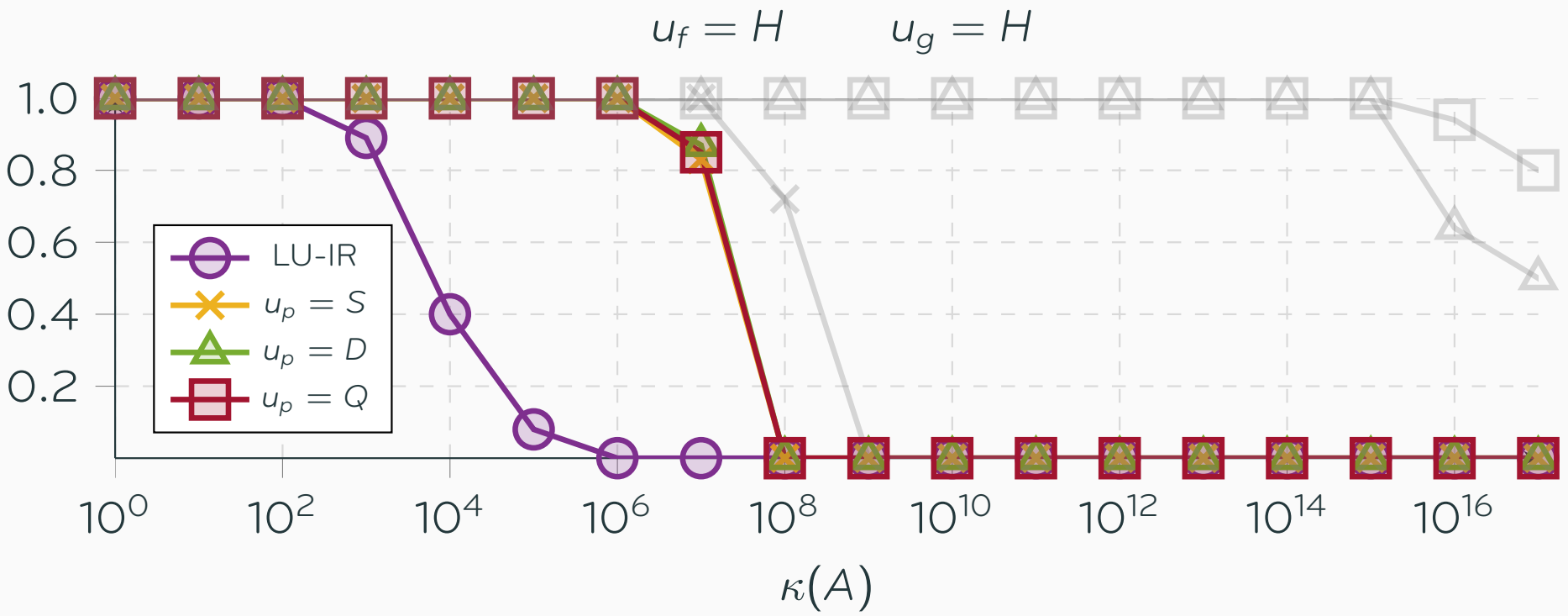
- GMRES-IR5 with $u_g = D$
  - if $u_p = Q$ (i.e., GMRES-IR3) 100% success up to $\kappa(A) \simeq 10^{16}$
  - if $u_p = D$ only little robustness loss but (potentially) much higher efficiency
  - if $u_p = S$ still much better than LU-IR3

Experiments : robustness over $\kappa(A)$



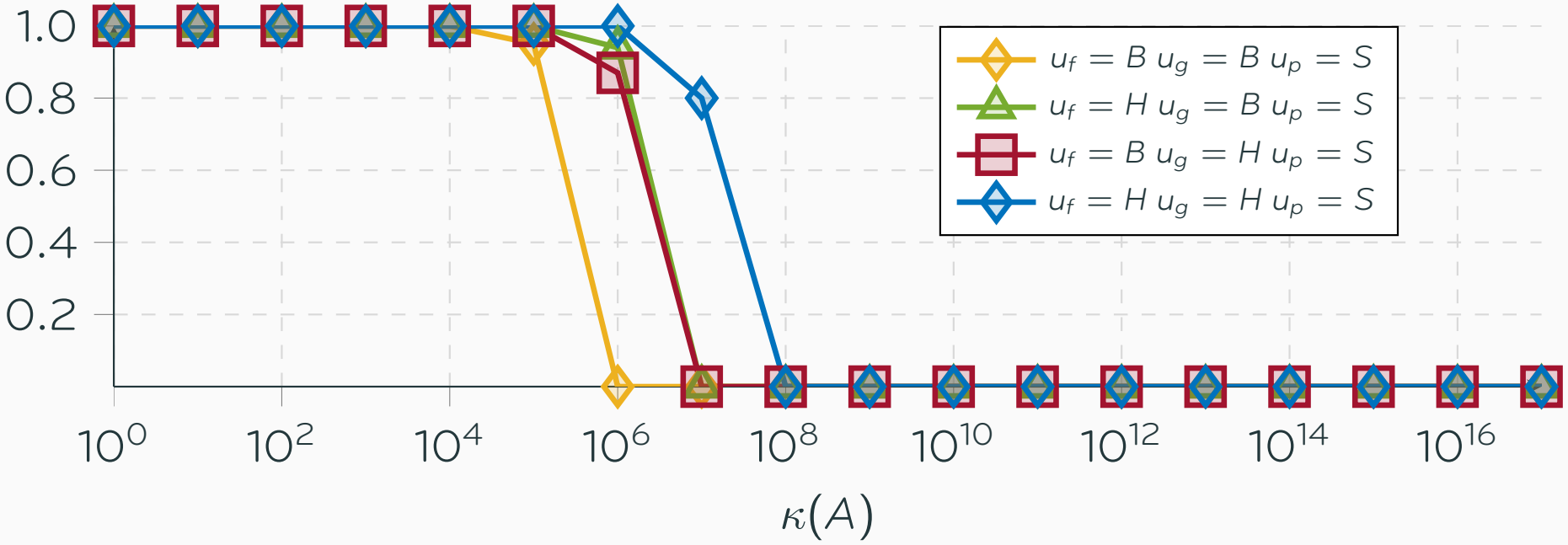
- GMRES-IR5 with $u_g = D \rightarrow S$
  - if $u_g = S$, $u_p = Q$ and $u_p = D$ same bounds $\Rightarrow$ $u_p = Q$ not relevant
  - if $u_p = S$, $u_g = D$ and $u_g = S$ same bounds $\Rightarrow$ $u_g = D$ not relevant

Experiments : robustness over $\kappa(A)$



- GMRES-IR5 with $u_g = D \rightarrow S \rightarrow H$
  - less robust than previous variants but much better than LU-IR3

Five-precisions combinations



2 **five-precision** combinations meaningful theoretically :

$(u_f = B, u = D, u_r = Q, u_g = H, u_p = S)$ $(u_f = H, u = D, u_r = Q, u_g = B, u_p = S)$

$\Rightarrow$ **Tradeoff** between 2 four-precision combinations allowing even finer setup of convergence conditions.

Sparse specific features of LU-IR3

Fill-in in sparse direct solvers : $\mathbf{nnz(A)} \ll \mathbf{nnz(LU)}$.

- $n \times n$ matrix from a regular 3D geometry $\Rightarrow \mathrm{nnz}(A) = \mathcal{O}(n)$ and $\mathrm{nnz}(LU) = \mathcal{O}(n^{4/3})$.
- Operations complexities : facto $\mathcal{O}(n^2)$, solve $\mathcal{O}(n^{4/3})$, SpMV $\mathcal{O}(n)$.

Sparse specific features of LU-IR3

Fill-in in sparse direct solvers : $\mathbf{nnz(A) \ll nnz(LU)}$.

- $n \times n$ matrix from a regular 3D geometry $\Rightarrow \mathrm{nnz}(A) = \mathcal{O}(n)$ and $\mathrm{nnz}(LU) = \mathcal{O}(n^{4/3})$.
- Operations complexities : facto $\mathcal{O}(n^2)$, solve $\mathcal{O}(n^{4/3})$, SpMV $\mathcal{O}(n)$.

Thus, sparse IR has **important specific features** :

- SpMV much cheaper than solve $\Rightarrow$ **$u_r \ll u$ has limited impact on time/memory performance** (even for $u_r =$ fp128).
- Memory space of A in $u_r \leq u$ negligible in comparison of the LU factors in $u_f \Rightarrow$ **LU-IR3 saves memory** on LU factors in u !
- For multifrontal sparse solver $\mathbf{mem(LU) \leq Mem\ Peak} \Rightarrow$ **GMRES-IR5 can save memory** even if the factors are casted in higher precision.

Implemented methods for double precision solution

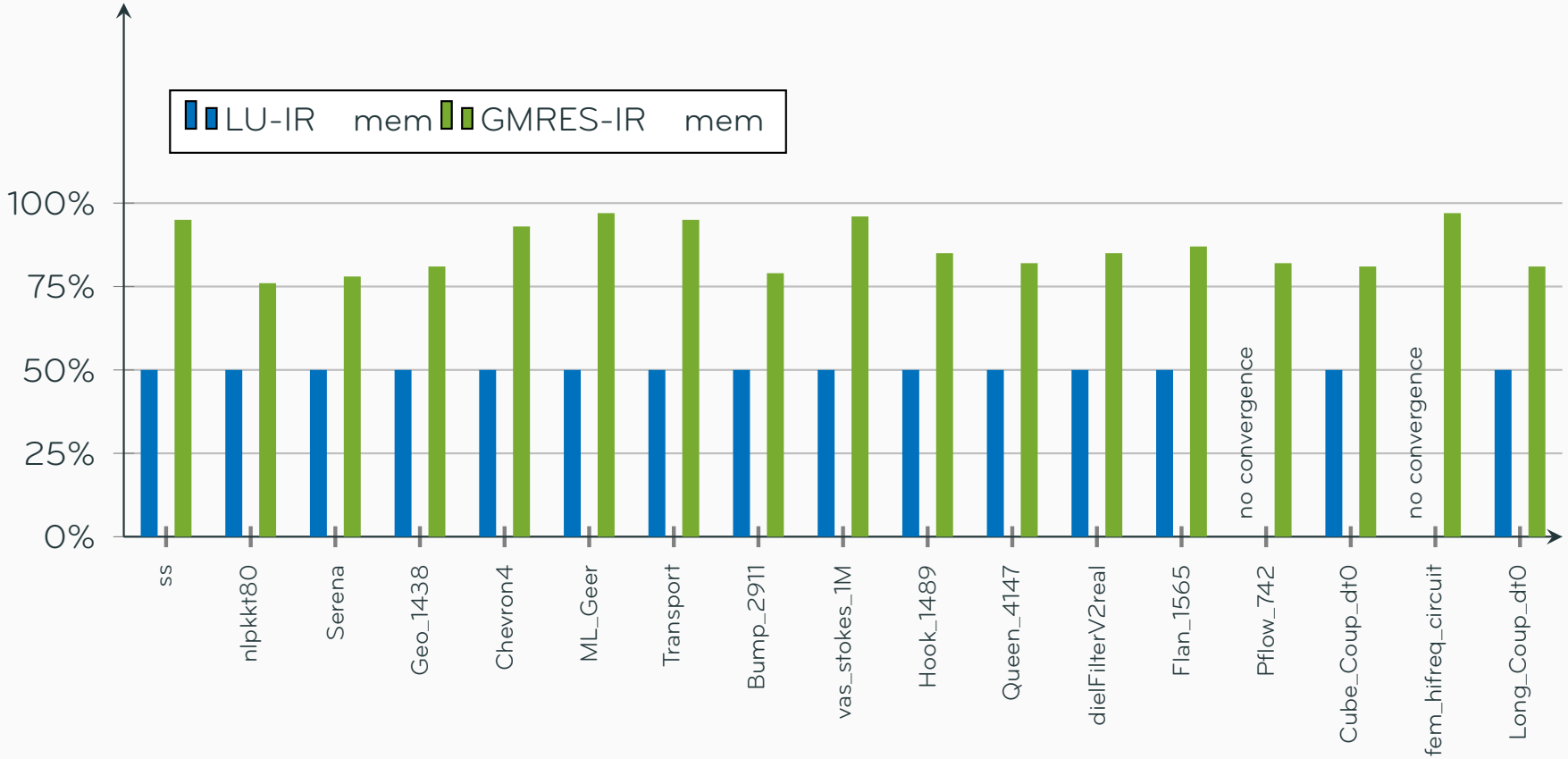
Solver	u_f	u	u_r	u_g	u_p	$\max(\kappa(A))$	ferr
DMUMPS	fp64 LU standard solver					–	$\kappa(A) \times 10^{-16}$
LU-IR	S	D	D	–	–	2×10^7	$\kappa(A) \times 10^{-16}$
GMRES-IR	S	D	D	D	D	1×10^{10}	$\kappa(A) \times 10^{-16}$
LU-IR+	S	D	Q	–	–	2×10^7	10^{-16}
GMRES-IR+	S	D	Q	D	D	1×10^{10}	10^{-16}

For the GMRES variants since $u_p \neq u_f$, the LU factors are casted.

All IR variants use single precision factorization to accelerate.

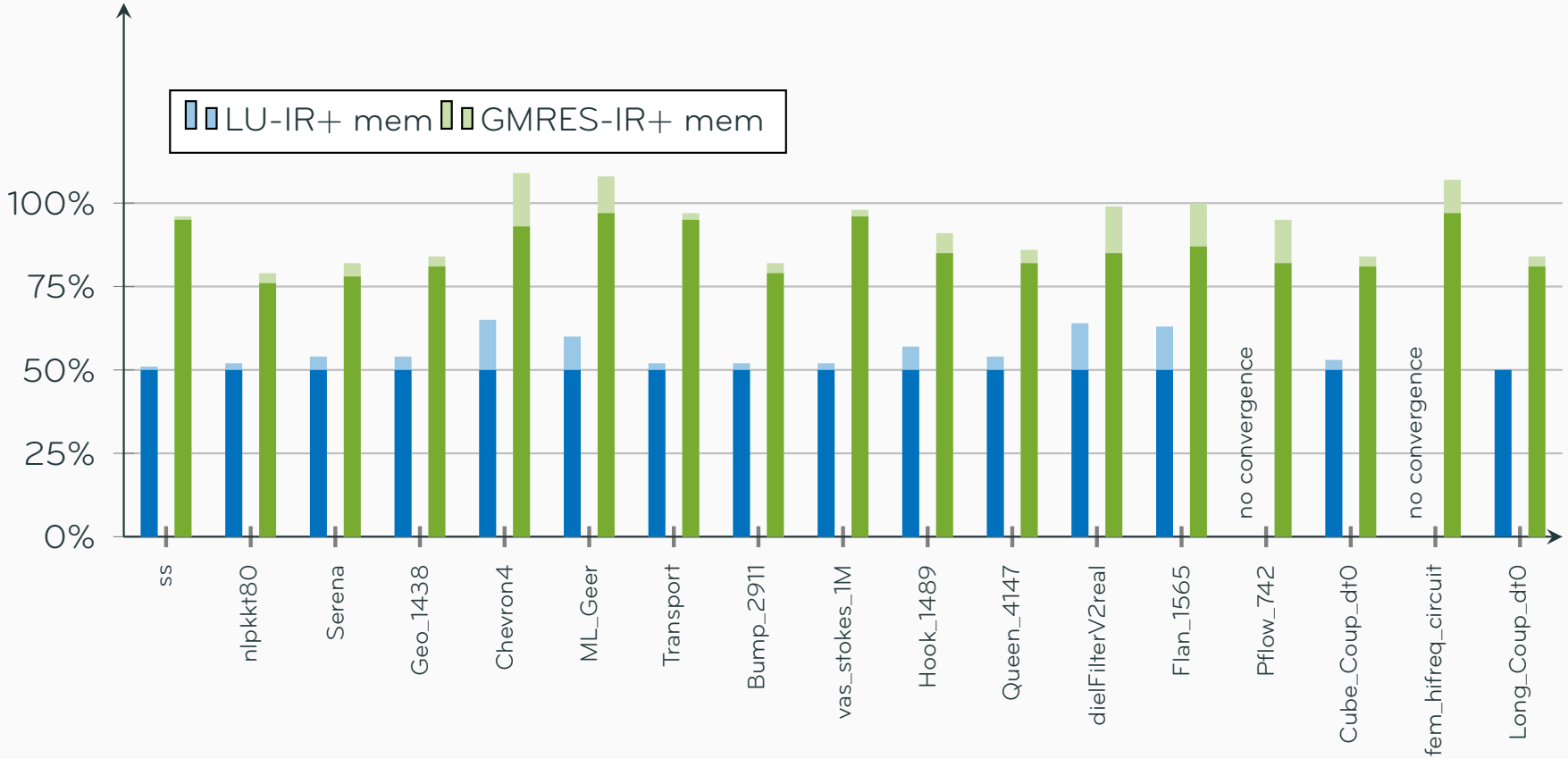
We use the multifrontal solver MUMPS for factorization and LU solve operations.

Memory Performance (No MPI - 18 threads)



- LU-IR consumes **2× less memory**!
- Mem Peak > nnz(LU) ⇒
  GMRES-IR gains at best 25%.

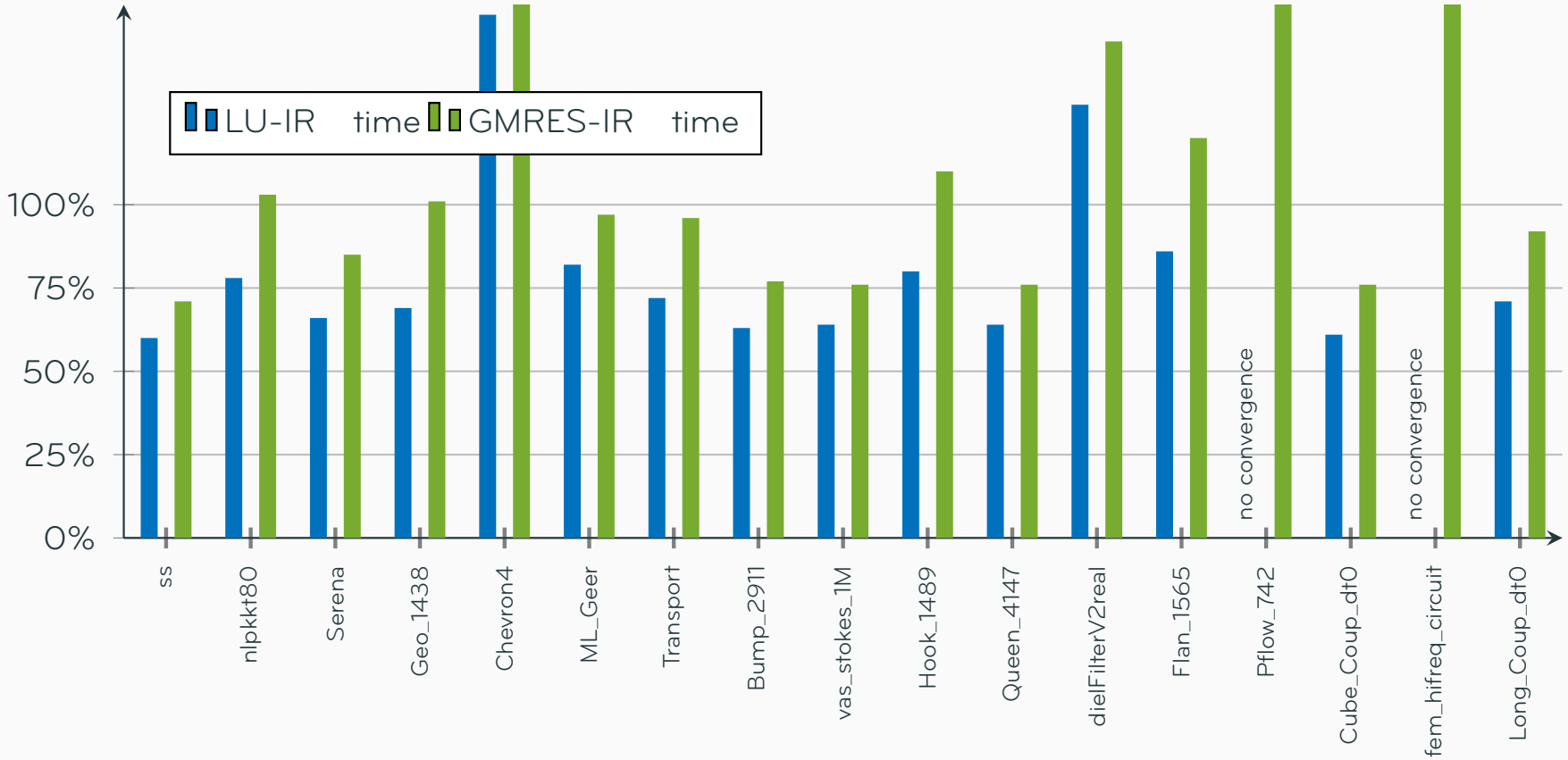
Memory Performance (No MPI - 18 threads)



- LU-IR consumes **2× less memory**!
- Mem Peak > nnz(LU) ⇒ **GMRES-IR gains** at best 25%.

- **Overhead** to reach a 10^{-16} ferr is on most matrices **negligible**.
- Heavy overhead when : **small matrices**, **small fill-in**.

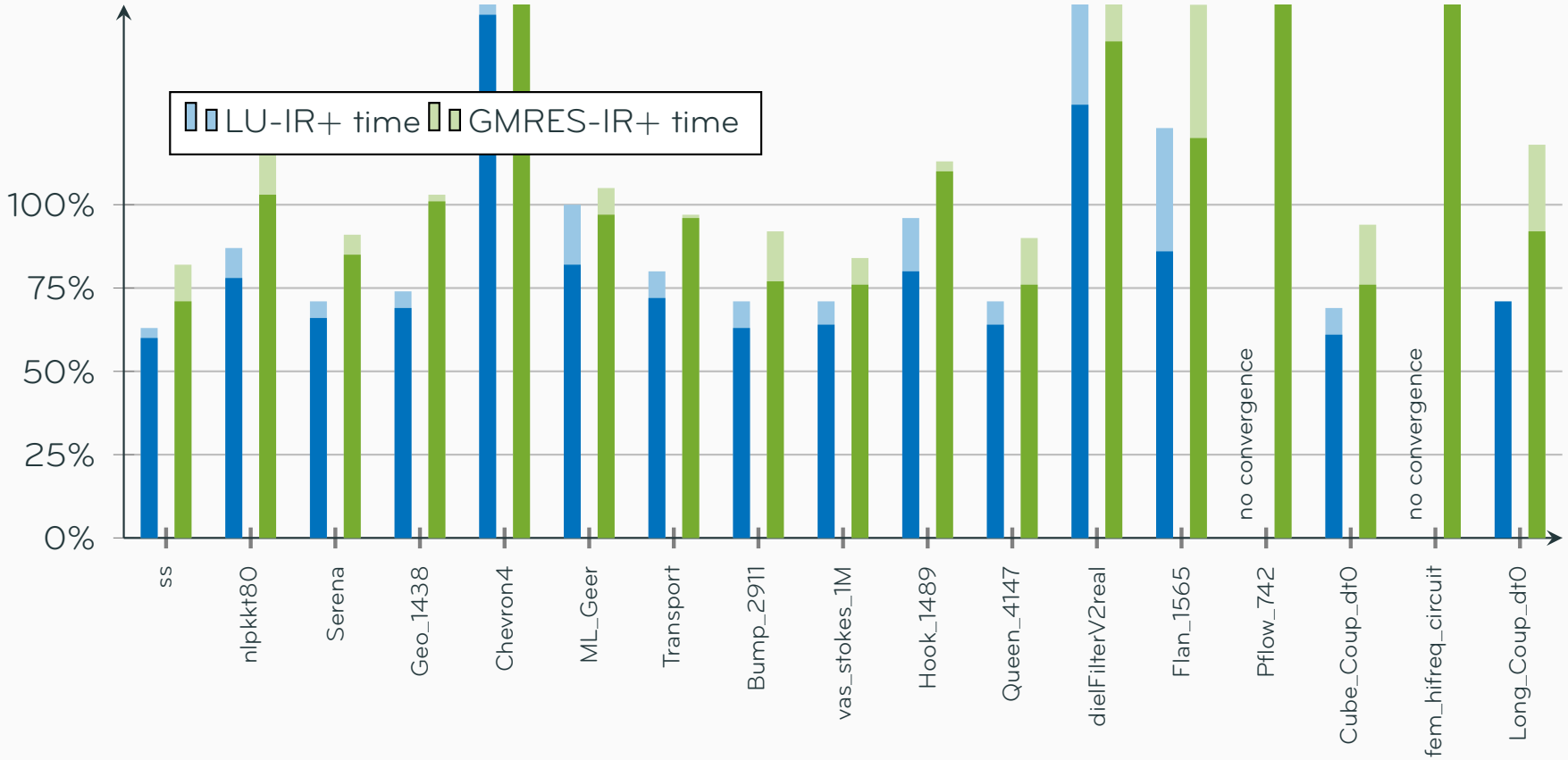
Time Performance (No MPI - 18 threads)



- LU-IR **1.4–1.7× quicker** on most of the matrices !
- GMRES-IR **slower** than LU-IR (ask more LU solves), but converges on all the matrices ⇒ **more robust**.

- Underperformance when : **small matrices**, **small fill-in**, **high number of iterations**.

Time Performance (No MPI - 18 threads)

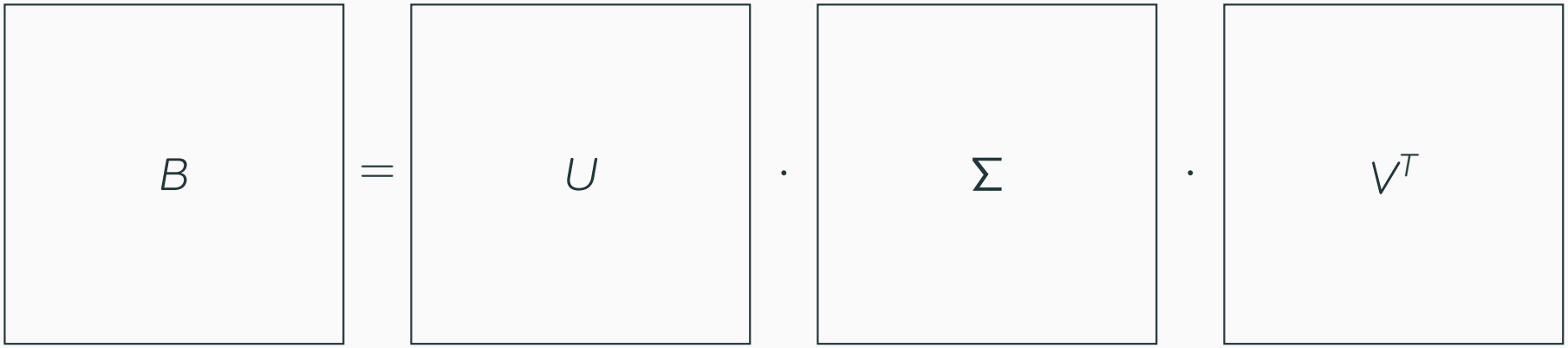


- LU-IR **1.4–1.7× quicker** on most of the matrices !
- GMRES-IR **slower** than LU-IR (more LU solves), but converges on all the matrices ⇒ **more robust**.
- Performance loss when : **small matrices**, **small fill-in**, **high number of iterations**.
- **Overhead** to reach a 10^{-16} ferr is on most matrices **reasonable**.

Mixed-precision block low-rank

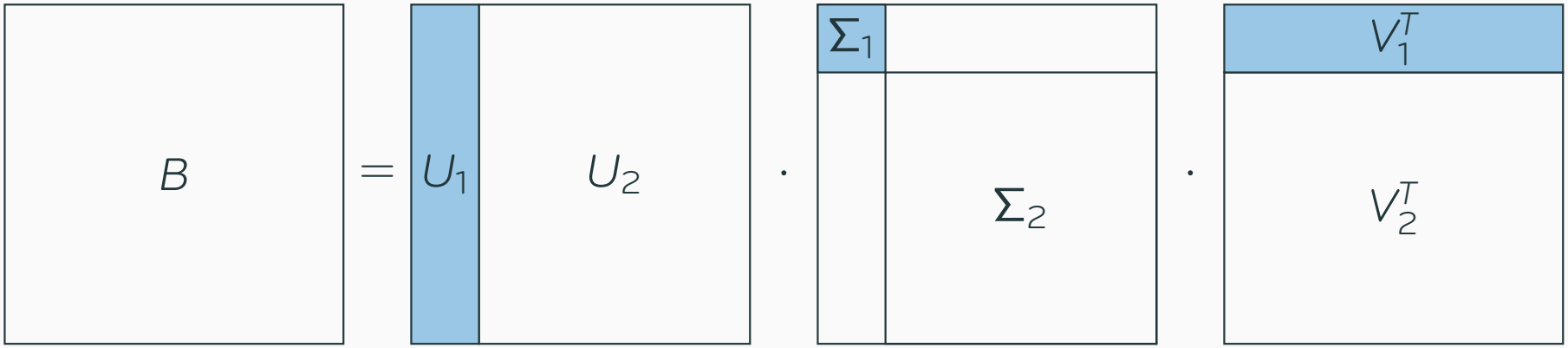
Low-rank approximation

Take a dense matrix B of size $b \times b$ and its SVD $B = USV^T$



Low-rank approximation

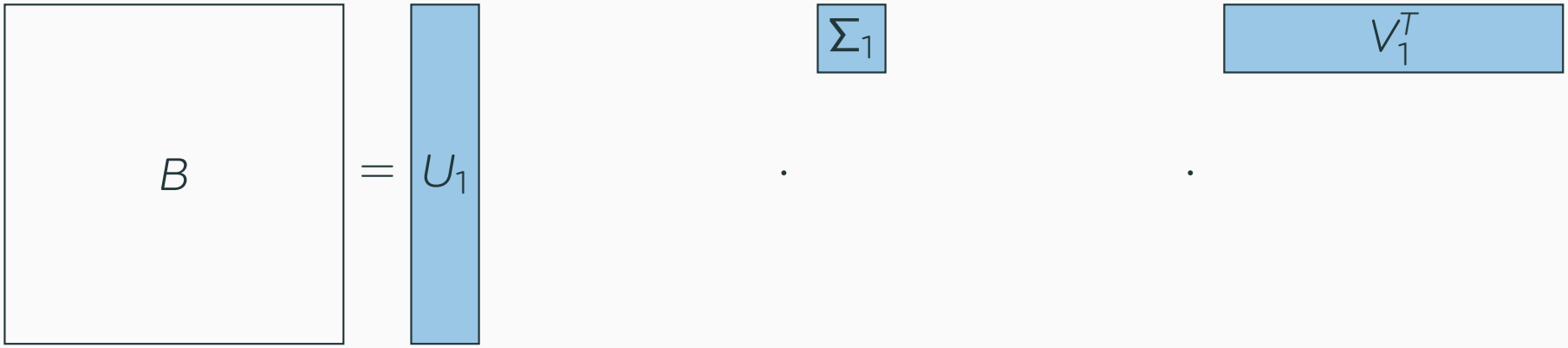
Take a dense matrix B of size $b \times b$ and its SVD $B = USV^T$



$$B = U_1\Sigma_1V_1^T + U_2\Sigma_2V_2^T \quad \text{with} \quad \Sigma_1(k,k) = \sigma_k > \varepsilon,\ \Sigma_2(1,1) = \sigma_{k+1} \leq \varepsilon$$

Low-rank approximation

Take a dense matrix B of size $b \times b$ and its SVD $B = USV^T$



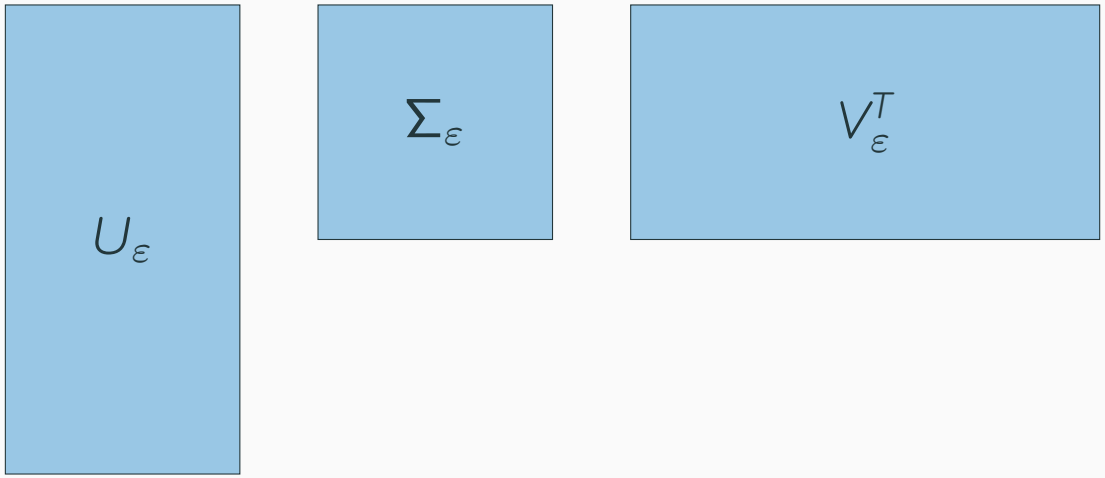
$$B = U_1\Sigma_1V_1^T + U_2\Sigma_2V_2^T \quad \text{with} \quad \Sigma_1(k,k) = \sigma_k > \varepsilon,\ \Sigma_2(1,1) = \sigma_{k+1} \leq \varepsilon$$

$$\text{If } \tilde{B} = U_1\Sigma_1V_1^T \quad \text{then} \quad \|B - \tilde{B}\|_2 = \|U_2\Sigma_2V_2^T\|_2 = \sigma_{k+1} \leq \varepsilon$$

If the singular values of B decay very fast (e.g. exponentially) then $k \ll b$ even for very small ε (e.g., 10^{-14}) $\rightarrow$ memory and CPU consumption can be reduced considerably with a controlled loss of accuracy ($\leq \varepsilon$) if $\tilde{B}$ is used instead of B

Mixed precision block low-rank

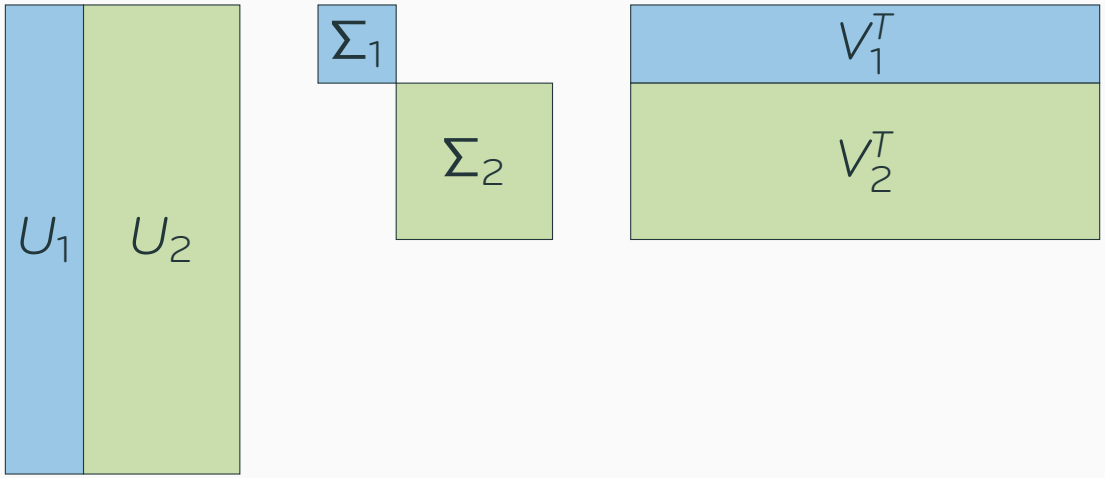
What precision should we use for storing a compressed block ?



- Naive answer : a precision whose roundoff is safely smaller than ε, e.g. if $\varepsilon = 10^{-9}$ use double precision

Mixed precision block low-rank

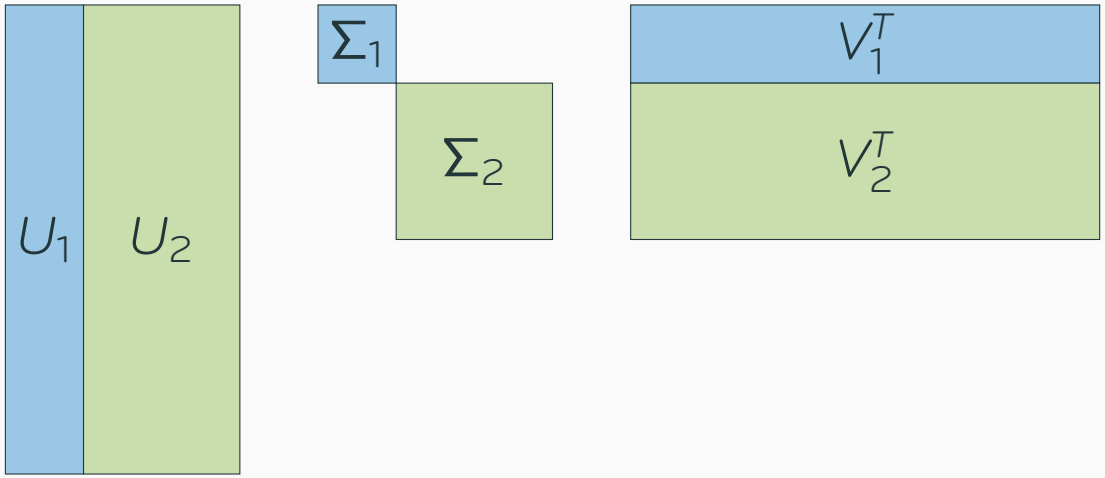
What precision should we use for storing a compressed block ?



- Naive answer : a precision whose roundoff is safely smaller than ε, e.g. if $\varepsilon = 10^{-9}$ use double precision
- Our idea : let $U_\varepsilon = [U_1 U_2]$, $\Sigma_\varepsilon = \mathrm{diag}(\Sigma_1 \Sigma_2)$, $V_\varepsilon = [V_1 V_2]$. Converting U_2 and V_2 to single precision introduces an error of order $u_{fp32} \|\Sigma_2\|$

Mixed precision block low-rank

What precision should we use for storing a compressed block ?

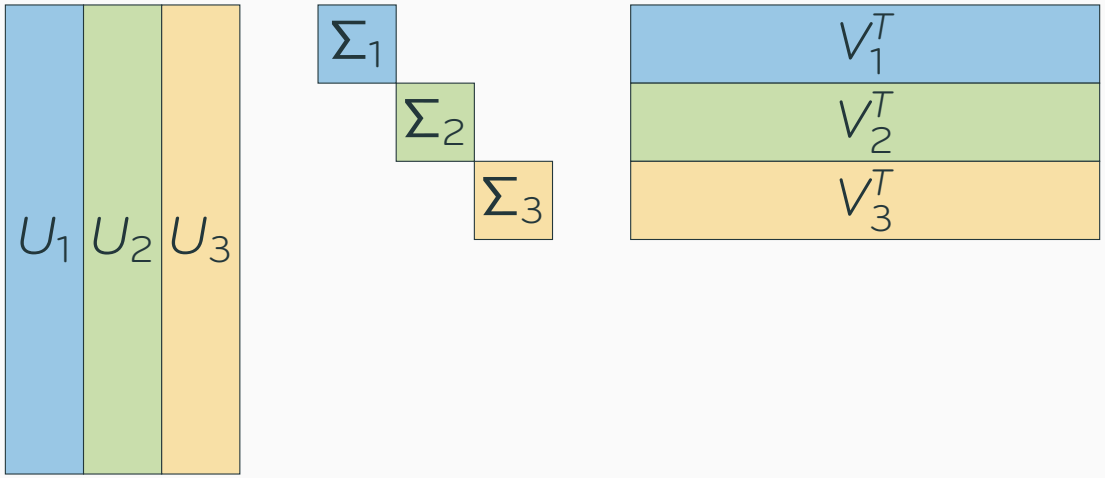


- Naive answer : a precision whose roundoff is safely smaller than ε, e.g. if $\varepsilon = 10^{-9}$ use double precision
- Our idea : let $U_\varepsilon = [U_1 U_2]$, $\Sigma_\varepsilon = \mathrm{diag}(\Sigma_1 \Sigma_2)$, $V_\varepsilon = [V_1 V_2]$. Converting U_2 and V_2 to single precision introduces an error of order $u_{fp32}\|\Sigma_2\|$

If $\|\Sigma_2\| \leq \varepsilon / u_{fp32} \sim 2 \times 10^{-2}$ error is $\leq \varepsilon \|A\|$

Mixed precision block low-rank

What precision should we use for storing a compressed block ?



- Naive answer : a precision whose roundoff is safely smaller than ε, e.g. if $\varepsilon = 10^{-9}$ use double precision
- Our idea : let $U_\varepsilon = [U_1 U_2]$, $\Sigma_\varepsilon = \mathrm{diag}(\Sigma_1 \Sigma_2)$, $V_\varepsilon = [V_1 V_2]$. Converting U_2 and V_2 to single precision introduces an error of order $u_{fp32}\|\Sigma_2\|$

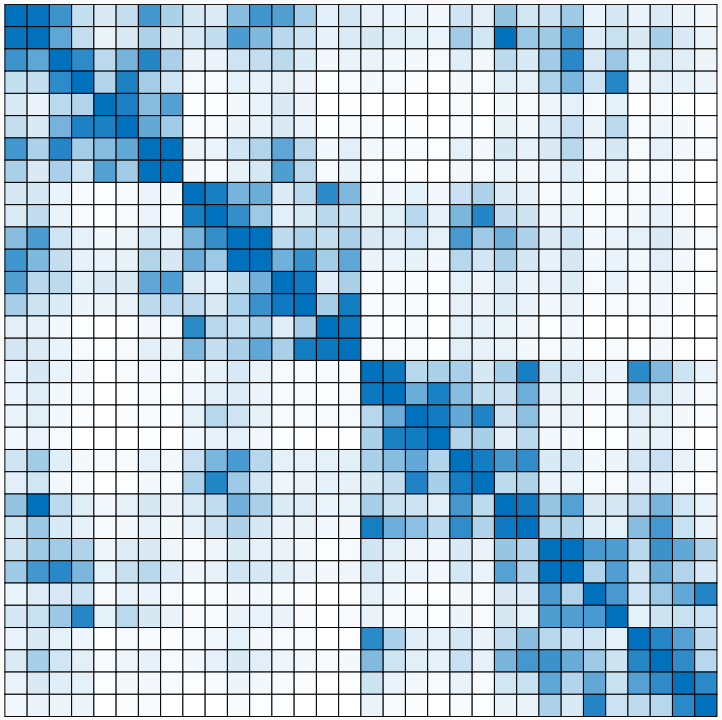
If $\|\Sigma_2\| \leq \varepsilon / u_{fp32} \sim 2 \times 10^{-2}$ error is $\leq \varepsilon \|A\|$

Can be generalized to p precisions $u_1 \leq \varepsilon \leq u_2 \leq \cdots \leq u_p$

Mixed precision block low-rank

Schur complement of a Poisson problem :

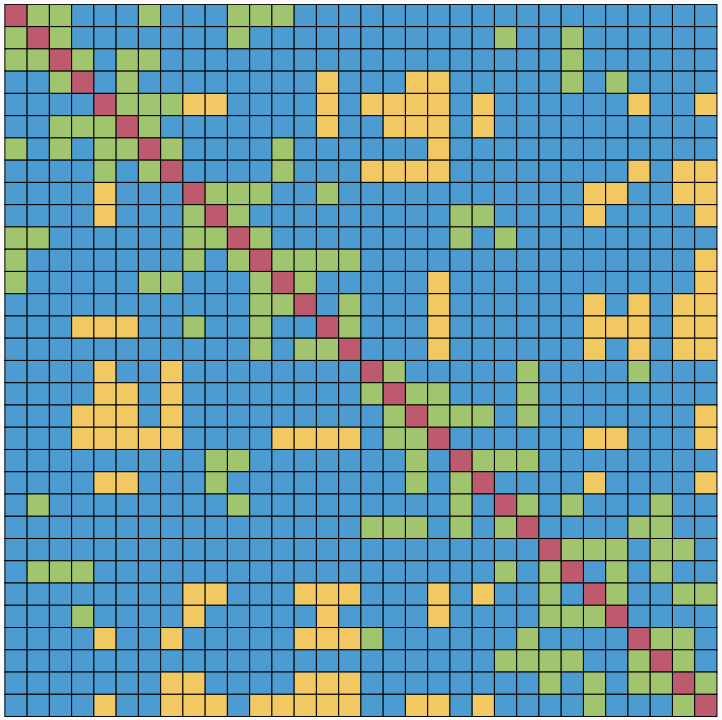
matrix size 4096, block size 128, $\varepsilon = 10^{-10}$



Mixed precision block low-rank

Schur complement of a Poisson problem :

matrix size 4096, block size 128, $\varepsilon = 10^{-10}$



- fp64
- fp64-fp32-bfloat16
- f32-bfloat16
- bfloat16

- Full double precision is required only for full-rank blocks
- Most blocks are stored in precision with roundoff $u \ll \varepsilon$

Mixed precision block low-rank : factorization

- Traditional LU (Wilkinson)

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

- Uniform precision BLR LU (Higham & Mary)

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

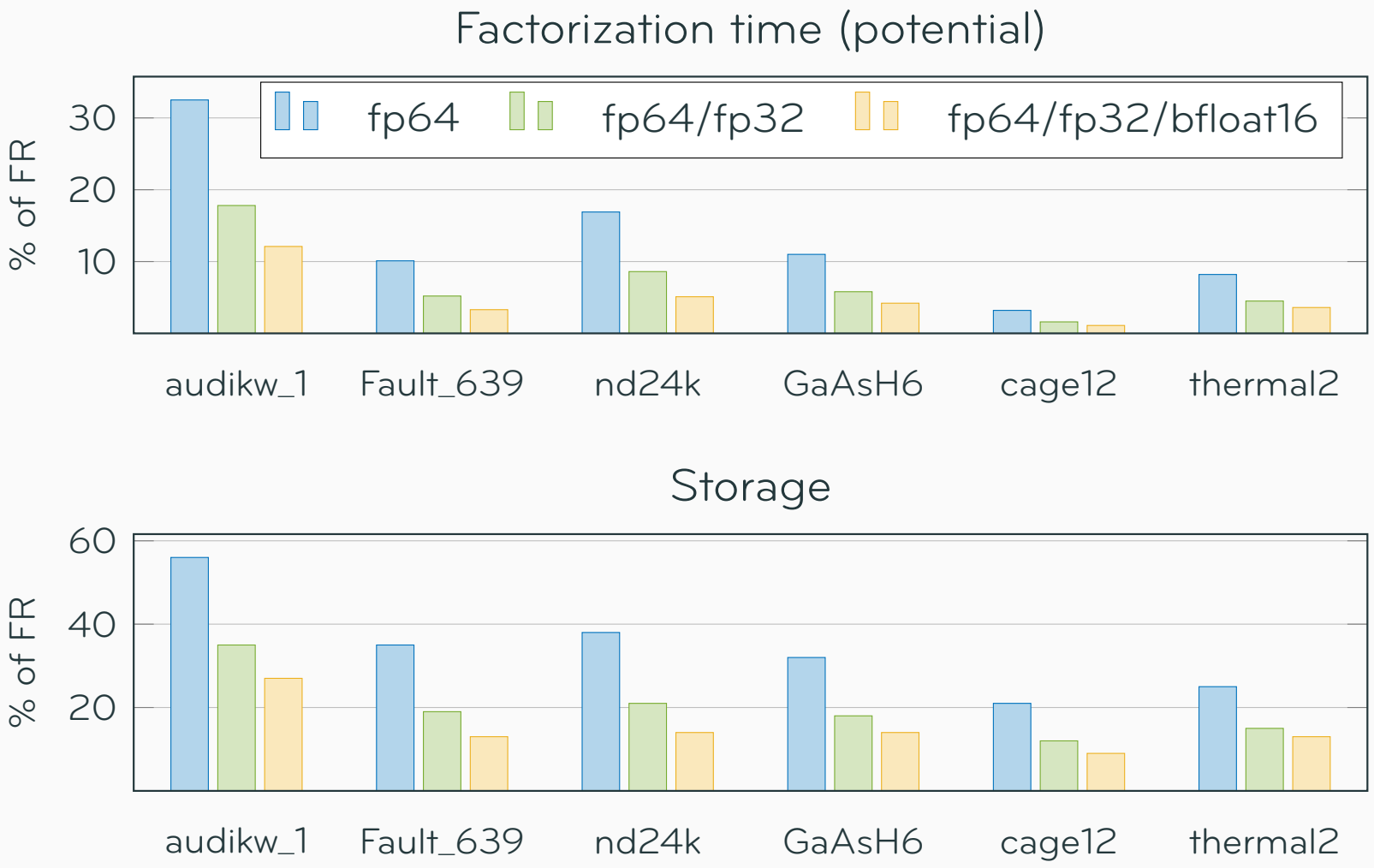
- Mixed precision BLR LU

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

with $c_1,\ c_2,\ c_1',\ c_2' \in O\left(n^2(b+q)\rho_n\right)$

Backward stability is preserved in the mixed precision block low-rank factorization

Mixed precision block low-rank : experimental results



Considerable gains with marginal error increase

Wrap up

Conclusions

- GMRES-based iterative refinement in five precisions is a versatile method offering many options for trading off performance and accuracy/robustness
- Using mixed precision in low-rank approximation leads to considerable storage and (potentially) performance improvements with practically no impact on the accuracy

Future work

- High-performance, parallel implementation of both methods on systems with half precision units
- Combine the two approaches
- Cast matrix and factors on the fly (see Artwig's upcoming talk)