Sparse Days in Saint-Girons IV

20-22 June 2022

Adaptive Precision Solvers for Sparse and Data Sparse Systems

Theo Mary

Sorbonne Université, CNRS, LIP6

Slides available at https://bit.ly/adapt2022

Solution of $Ax = b$, A large and sparse:

- **Direct methods**
  - Robust, black box solvers
  - High time and memory cost for factorization of A

- **Iterative methods**
  - Low time and memory per-iteration cost
  - Convergence is application dependent

Solution of $Ax = b$, A large and sparse:

- **Direct methods**
  - Robust, black box solvers
  - High time and memory cost for factorization of A

  $\Rightarrow$ Need fast factorization

- **Iterative methods**
  - Low time and memory per-iteration cost
  - Convergence is application dependent

  $\Rightarrow$ Need good preconditioner

Solution of $Ax = b$, A large and sparse:

- **Direct methods**
  - Robust, black box solvers
  - High time and memory cost for factorization of A
  - $\Rightarrow$ Need fast factorization

- **Iterative methods**
  - Low time and memory per-iteration cost
  - Convergence is application dependent
  - $\Rightarrow$ Need good preconditioner

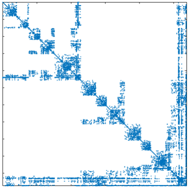
$\Rightarrow$ **Approximate factorizations...**

- as approximate fast direct methods, if
  - low accuracy is sufficient, or
  - matrix is structured (data sparsity)
- as high quality preconditioners otherwise

Dropping approximations (sparsification)

Dropping: replace with zero any value sufficiently small

$$|a_{ij}| \leq \epsilon \|A\| \quad \Rightarrow \quad a_{ij} \leftarrow 0$$

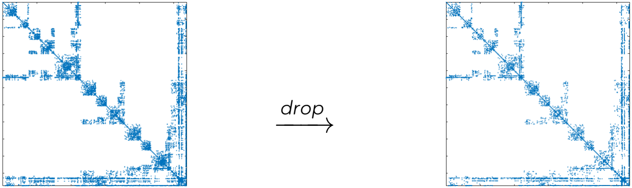


sparse A

Dropping approximations (sparsification)

Dropping: replace with zero any value sufficiently small

$$|a_{ij}| \leq \epsilon \|A\| \quad \Rightarrow \quad a_{ij} \leftarrow 0$$

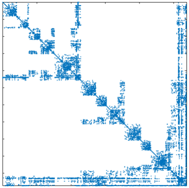


sparse A sparser A

Dropping approximations (sparsification)

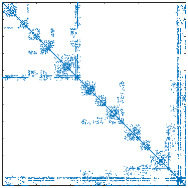
Dropping: replace with zero any value sufficiently small

$$\begin{cases} |\ell_{ij} u_{jj}| \leq \epsilon \|A\| & \Rightarrow \quad \ell_{ij} \leftarrow 0 \\ |u_{ij}| \leq \epsilon \|A\| & \Rightarrow \quad u_{ij} \leftarrow 0 \end{cases}$$

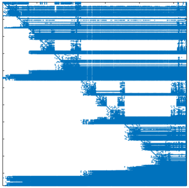


sparse A

$\xrightarrow{\textit{drop}}$

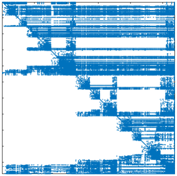


sparser A



LU factors

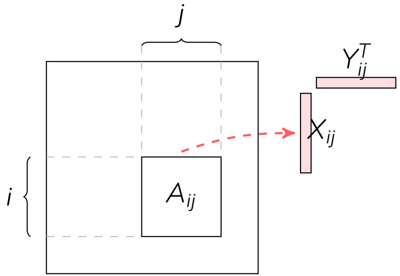
$\xrightarrow{\textit{drop}}$

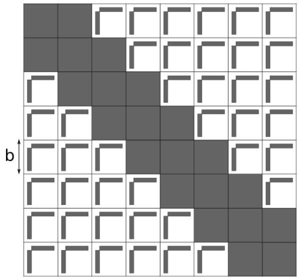


incomplete LU

Low-rank approximations (data sparsification)

Low-rank compression: given $A = U\Sigma V^T$, if we truncate singular vectors associated with $\sigma_i \leq \epsilon$, we obtain $\widetilde{A}$ such that $\|\widetilde{A} - A\| \leq \epsilon$





Block Low Rank

Compress A_{ij} such that $\|\widetilde{A}_{ij} - A_{ij}\| \leq \epsilon\|A\|$:

- If $\|A_{ij}\| \leq \epsilon\|A\| \Rightarrow A_{ij} \leftarrow 0$ (drop block)
- otherwise replace A_{ij} with $\widetilde{A}_{ij} = X_{ij}Y_{ij}^T$

Evolution of the floating-point landscape

Common point: these methods only deal in absolutes: either we keep the data at full accuracy, or we discard it completely!

Evolution of the floating-point landscape

Common point: these methods only deal in absolutes: either we keep the data at full accuracy, or we discard it completely!

		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}

Evolution of the floating-point landscape

Common point: these methods only deal in absolutes: either we keep the data at full accuracy, or we discard it completely!

		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}

We need **a new paradigm** that uses multiple, gradual levels of approximation

Mixed precision algorithms

Acta Numerica (2022), pp. 347–414
doi:10.1017/S0962492922000022

Mixed precision algorithms in numerical linear algebra

Nicholas J. Higham
Department of Mathematics, University of Manchester, Manchester, M13 9PL, UK
E-mail: nick.higham@manchester.ac.uk

Theo Mary
Sorbonne Université, CNRS, LIP6, Paris, F-75005, France
E-mail: theo.mary@lip6.fr

https://bit.ly/mixed-survey

Acta Numerica (2022), pp. 347–414
doi:10.1017/S0962492922000022

Mixed precision algorithms in numerical linear algebra

Nicholas J. Higham
Department of Mathematics, University of Manchester, Manchester, M13 9PL, UK
E-mail: nick.higham@manchester.ac.uk

Theo Mary
Sorbonne Université, CNRS, LIP6, Paris, F-75005, France
E-mail: theo.mary@lip6.fr

https://bit.ly/mixed-survey

Adaptive precision algorithms: an emerging subclass

- Anzt, Dongarra, Flegar, Higham, and Quintana-Orti, *Adaptive precision in block-Jacobi preconditioning for iterative sparse linear system solvers* (2019).
- Doucet, Ltaief, Gratadour, and Keyes, *Mixed-precision tomographic reconstructor computations on hardware accelerator* (2019).
- Ahmad, Sundar, and Hall, *Data-driven mixed precision sparse matrix vector multiplication for GPUs* (2019).
- Ooi, Iwashita, Fukaya, Ida, and Yokota, *Effect of mixed precision computing on H-matrix vector multiplication in BEM analysis* (2020).
- Diffenderfer, Osei-Kuffuor, and Menon, *QDOT: Quantized dot product kernel for approximate high-performance computing* (2021).
- Abdulah, Cao, Pei, Bosilca, Dongarra, Genton, Keyes, Ltaief, and Sun, *Accelerating geostatistical modeling and prediction with mixed-precision computations* (2022).

Adaptive precision algorithms

- Given an algorithm and a prescribed accuracy ϵ, employ the minimal precision for each instruction

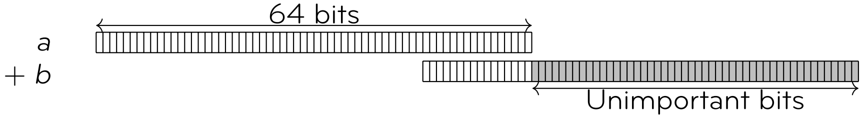
$\Rightarrow$ **First of all, why should the precisions vary?**

Adaptive precision algorithms

- Given an algorithm and a prescribed accuracy ϵ, employ the minimal precision for each instruction

$\Rightarrow$ **First of all, why should the precisions vary?**

- Because not all computations are equally "important"!
  Example:



$\Rightarrow$ **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

Adaptive precision SpMV

Graillat, Jézéquel, M., Molina (2022)

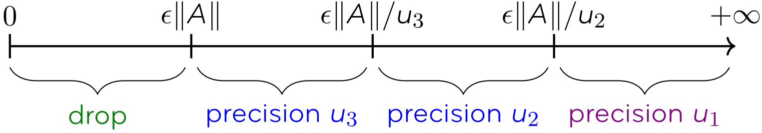
- **Goal:** compute the SpMV $y = Ax$ with accuracy ϵ using q precisions $u_1 \le \epsilon < u_2 < \ldots < u_q$
- Split elements a_{ij} on each row i into q buckets $B_{i1}, \ldots, B_{iq}$, where bucket B_{ik} uses precision u_k

Adaptive precision SpMV

Graillat, Jézéquel, M., Molina (2022)

- **Goal:** compute the SpMV $y = Ax$ with accuracy ϵ using q precisions $u_1 \le \epsilon < u_2 < \ldots < u_q$
- Split elements a_{ij} on each row i into q buckets $B_{i1}, \ldots, B_{iq}$, where bucket B_{ik} uses precision u_k
- How should we build the buckets?

$$\begin{cases} |a_{ij}| \le \epsilon\|A\| & \Rightarrow \text{drop} \\ |a_{ij}| \in [\epsilon\|A\|/u_{k+1}, \epsilon\|A\|/u_k) & \Rightarrow \text{place in } B_{ik} \\ |a_{ij}| > \epsilon\|A\|/u_2 & \Rightarrow \text{place in } B_{i1} \end{cases}$$

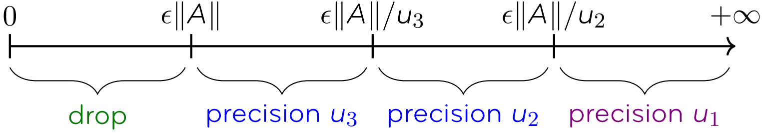


Adaptive precision SpMV

Graillat, Jézéquel, M., Molina (2022)

- **Goal:** compute the SpMV $y = Ax$ with accuracy ϵ using q precisions $u_1 \le \epsilon < u_2 < \ldots < u_q$
- Split elements a_{ij} on each row i into q buckets $B_{i1}, \ldots, B_{iq}$, where bucket B_{ik} uses precision u_k
- How should we build the buckets?

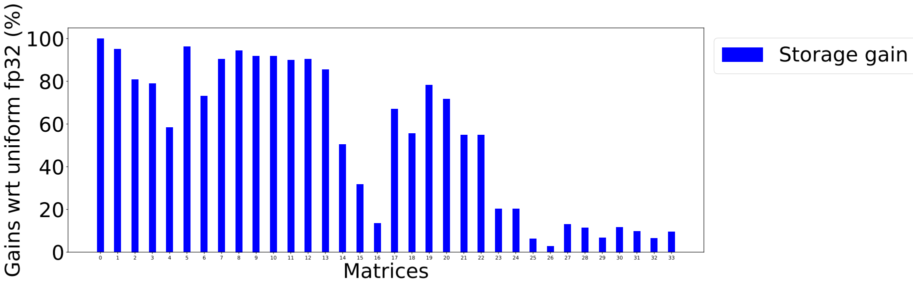
$$\begin{cases} |a_{ij}| \le \epsilon \|A\| & \Rightarrow \quad \text{drop} \\ |a_{ij}| \in [\epsilon\|A\|/u_{k+1}, \epsilon\|A\|/u_k) & \Rightarrow \quad \text{place in } B_{ik} \\ |a_{ij}| > \epsilon\|A\|/u_2 & \Rightarrow \quad \text{place in } B_{i1} \end{cases}$$



- **Theorem**: the computed $\widehat{y}$ satisfies $\|\widehat{y} - y\| \le c\epsilon\|A\|\|x\|$

Adaptive precision SpMV

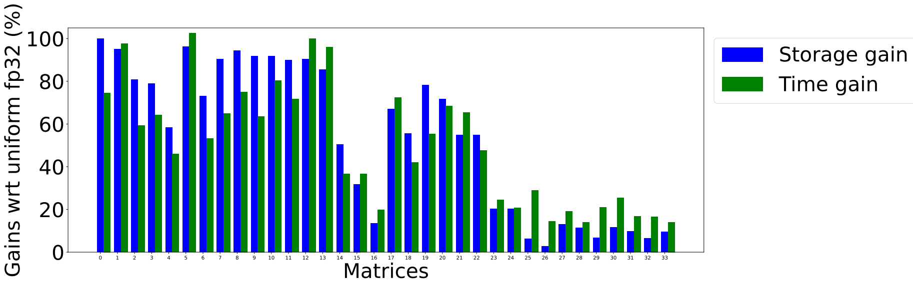
- 34 matrices from SuiteSparse of order $47k$–$11M$
- Timings on 24-core computer



Up to $36\times$ storage reduction

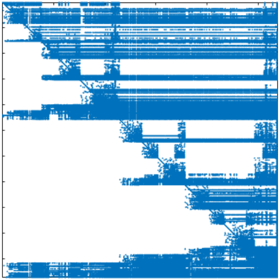
Adaptive precision SpMV

- 34 matrices from SuiteSparse of order $47k$–$11M$
- Timings on 24-core computer

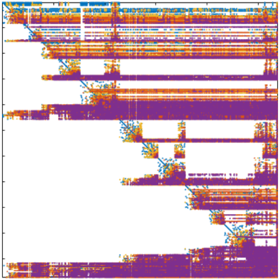


Up to $36\times$ storage reduction $\Rightarrow$ up to $7\times$ time reduction

Extension to adaptive precision ILU

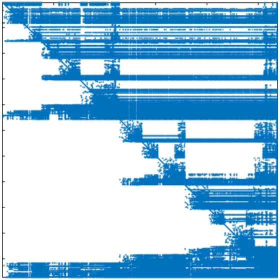


Incomplete LU
$\epsilon = 4 \times 10^{-7}$
storage$(L + U) = 81k$
$\kappa(U^{-1}L^{-1}A) = 60$

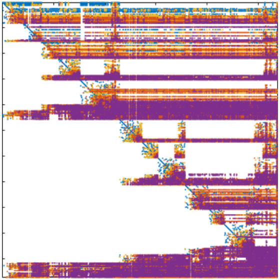


Adaptive LU
$\epsilon = 4 \times 10^{-7}$
storage$(L + U) = 43k$
$\kappa(U^{-1}L^{-1}A) = 60$

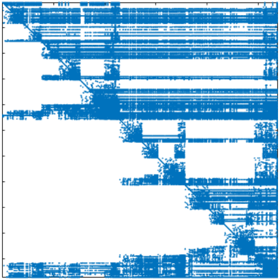
Extension to adaptive precision ILU



Incomplete LU
$\epsilon = 4 \times 10^{-7}$
$\text{storage}(L + U) = 81k$
$\kappa(U^{-1}L^{-1}A) = 60$

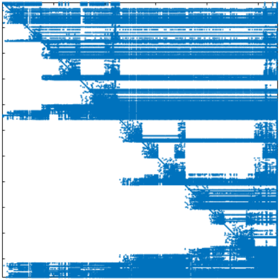


Adaptive LU
$\epsilon = 4 \times 10^{-7}$
$\text{storage}(L + U) = 43k$
$\kappa(U^{-1}L^{-1}A) = 60$

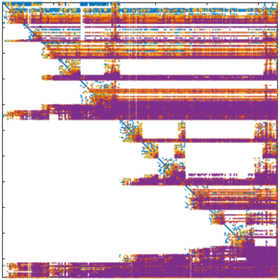


Incomplete LU
$\epsilon = 6 \times 10^{-5}$
$\text{storage}(L + U) = 43k$
$\kappa(U^{-1}L^{-1}A) = 2 \times 10^{7}$

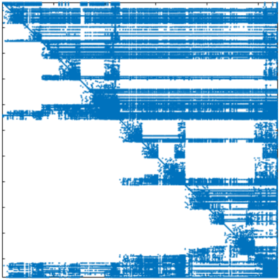
Extension to adaptive precision ILU



Incomplete LU
$\epsilon = 4 \times 10^{-7}$
$\text{storage}(L + U) = 81k$
$\kappa(U^{-1}L^{-1}A) = 60$



Adaptive LU
$\epsilon = 4 \times 10^{-7}$
$\text{storage}(L + U) = 43k$
$\kappa(U^{-1}L^{-1}A) = 60$

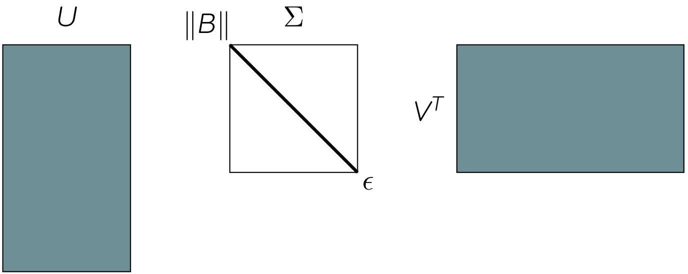


Incomplete LU
$\epsilon = 6 \times 10^{-5}$
$\text{storage}(L + U) = 43k$
$\kappa(U^{-1}L^{-1}A) = 2 \times 10^{7}$

Unlike SpMV, practical implementation seems challenging...
(future work)

Adaptive precision low rank compression

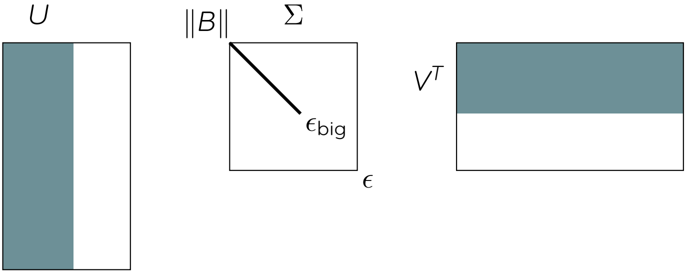
Amestoy, Boiteau, Buttari, Gerest, Jézéquel, L'Excellent, M. (2021)



How to increase low-rank compression?

Adaptive precision low rank compression

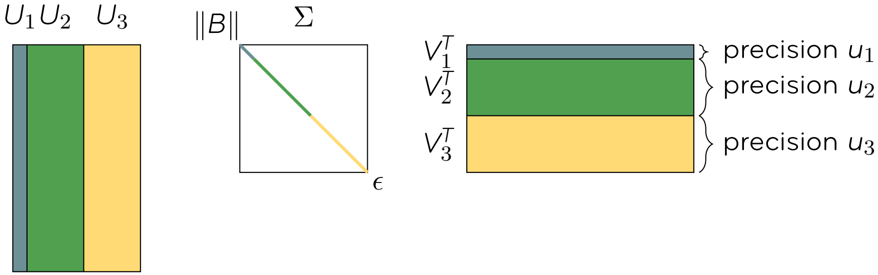
Amestoy, Boiteau, Buttari, Gerest, Jézéquel, L'Excellent, M. (2021)



How to increase low-rank compression?

- Standard approach: increase ϵ to discard more vectors

Amestoy, Boiteau, Buttari, Gerest, Jézéquel, L'Excellent, M. (2021)

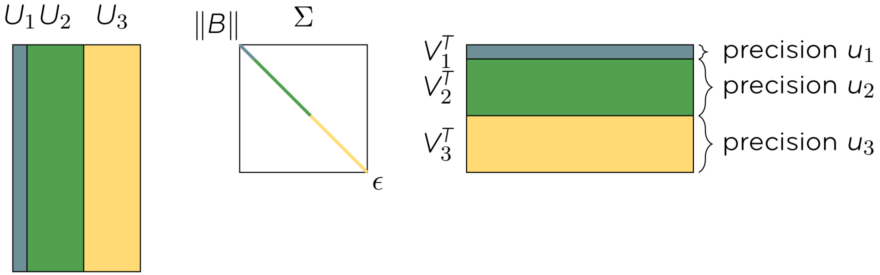


How to increase low-rank compression?

- Standard approach: increase ϵ to discard more vectors
- **Adaptive precision compression:** partition U and V into q groups of decreasing precisions $u_1 \le \epsilon < u_2 < \ldots < u_q$

Adaptive precision low rank compression

Amestoy, Boiteau, Buttari, Gerest, Jézéquel, L'Excellent, M. (2021)

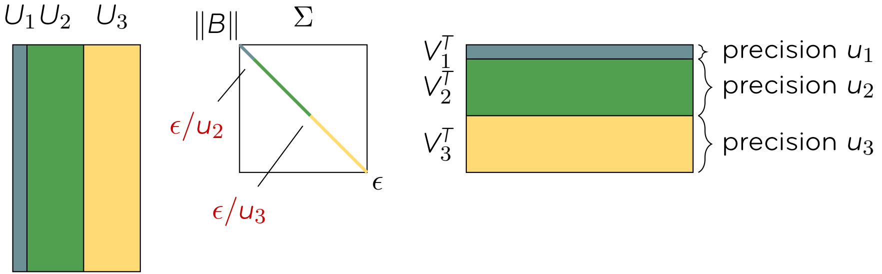


How to increase low-rank compression?

- Standard approach: increase ϵ to discard more vectors
- **Adaptive precision compression:** partition U and V into q groups of decreasing precisions $u_1 \le \epsilon < u_2 < \ldots < u_q$
- Why does it work? $B = \mathbf{B_1} + \mathbf{B_2} + \mathbf{B_3}$ with $|B_i| \le O(\|\Sigma_i\|)$

Adaptive precision low rank compression

Amestoy, Boiteau, Buttari, Gerest, Jézéquel, L'Excellent, M. (2021)

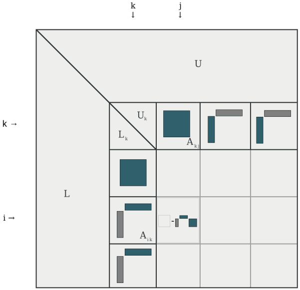


How to increase low-rank compression?

- Standard approach: increase ϵ to discard more vectors
- **Adaptive precision compression:** partition U and V into q groups of decreasing precisions $u_1 \le \epsilon < u_2 < \ldots < u_q$
- Why does it work? $B = \mathbf{B_1} + \mathbf{B_2} + \mathbf{B_3}$ with $|B_i| \le O(\|\Sigma_i\|)$

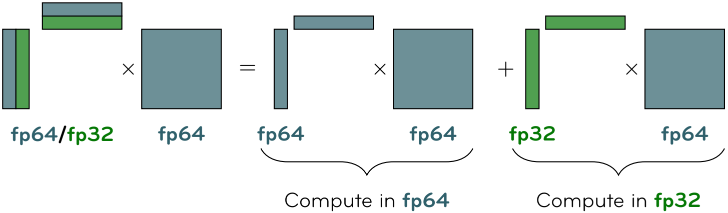
Adaptive precision BLR LU factorization

- Step k:
  - Compute $L_{kk}U_{kk} = A_{kk}$
  - Update
    $A_{ij} \leftarrow A_{ij} - (A_{ik}U_{kk}^{-1}) \times (L_{kk}^{-1}A_{kj})$



- Error analysis to both
  - prove stability: $\|A - \widehat{L}\widehat{U}\| \leq O(\epsilon)\|A\|$
  - **determine which precision is needed for each flop**

Example of kernel: LR × matrix multiplication:



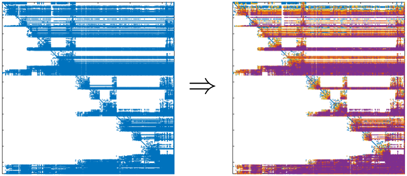
Adaptive precision BLR: results

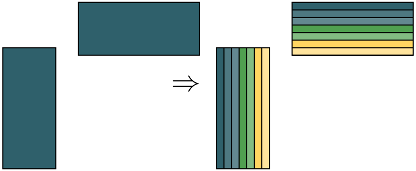
- Implementation within BLR multifrontal solver **MUMPS**
- Used to reduce storage and communications only for now (ongoing work to accelerate factorization)
- **7 precisions:** 16, 24, 32, 40, 48, 56, and 64 bits
- Using 8 MPI $\times$ 9 OpenMP (16×9 for knuckle)

Matrix		Factor size (GB)	Memory peak (GB)	Comm. vol. (GB)
thmgaz	BLR	95	120	5.5
	adapt. BLR	59	86	2.9
rubber	BLR	95	261	105.9
	adapt. BLR	66	220	69.8
knuckle	BLR	120	312	144.5
	adapt. BLR	71	259	70.7

Up to $1.7\times$ storage and $2.0\times$ comm. volume reductions

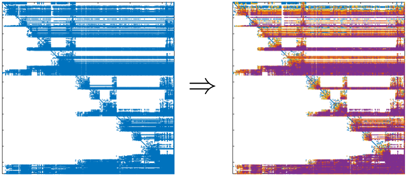
Take-home picture

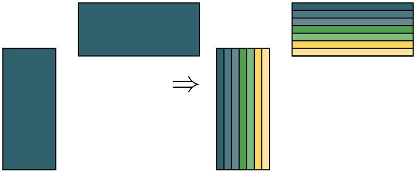




We now live in a multiprecision world,
we need to rethink our algorithms accordingly

Take-home picture





We now live in a multiprecision world,
we need to rethink our algorithms accordingly

Slides at https://bit.ly/adapt2022
Check out our papers:
  Adaptive SpMV: https://bit.ly/adapt2022-SpMV
  Adaptive BLR: https://bit.ly/adapt2022-BLR

Thank you!