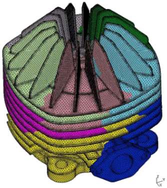
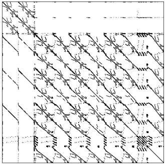
Accuracy and Stability of Block Low-Rank Linear Solvers

Theo Mary
University of Manchester, School of Mathematics

LIP6, Sorbonne Université, 6 December 2018

M\cr NA
Manchester Numerical Analysis

Context

 →  → 

Linear system $Ax = b$

Often a keystone in scientific computing applications
(discretization of PDEs, step of an optimization method, ...)

Large, sparse matrices

Matrix A is sparse (many zeros) but also large (10^6–10^9 unknowns)

Direct methods

Factorize $A = LU$ and solve $LUx = b$

☺ Numerically reliable ☹ Computational cost

Outline

1. **Complexity and performance of BLR linear solvers**

> P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *On the Complexity of the Block Low-Rank Multifrontal Factorization*. SIAM J. Sci. Comput. (2017).

> P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures*. ACM Trans. Math. Soft. (2018).

> P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *Bridging the gap between flat and hierarchical low-rank matrix formats: the multilevel BLR format*. Submitted (2018).

2. **Rounding error analysis of BLR factorization**
3. **Low-accuracy BLR preconditioners**

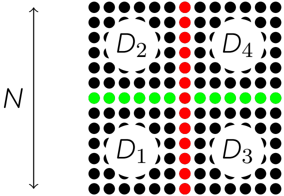
> N. Higham and T. Mary. *A New Preconditioner that Exploits Low-Rank Approximations to Factorization Error*. SIAM J. Sci. Comp (2018).

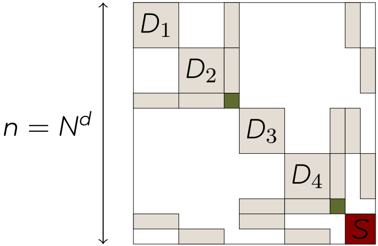
4. **Probabilistic rounding error analysis**

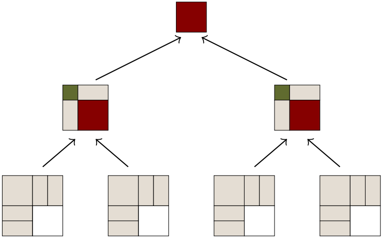
> N. Higham and T. Mary. *A New Approach to Probabilistic Rounding Error Analysis*. Submitted (2018).

Complexity and performance of BLR linear solvers

Structural sparsity







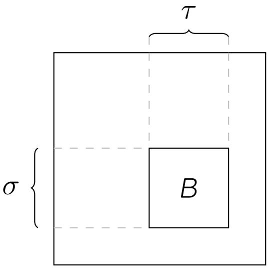
2D problem complexity

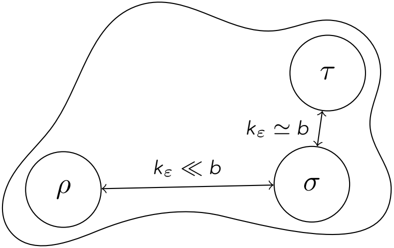
- Flops: $O(n^3) \to O(n^{3/2})$
- Storage: $O(n^2) \to O(n \log n)$

3D problem complexity

- Flops: $O(n^3) \to O(n^2)$
- Storage: $O(n^2) \to O(n^{4/3})$

In many cases of interest the matrix has a block low-rank structure



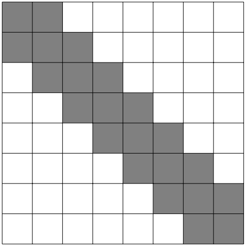


A block B represents the interaction between two subdomains.
Far away subdomains $\Rightarrow$ block of low numerical rank:

$$\begin{array}{cccc} B & \approx & X & Y^T \\ b \times b & & b \times k_\varepsilon & k_\varepsilon \times b \end{array}$$

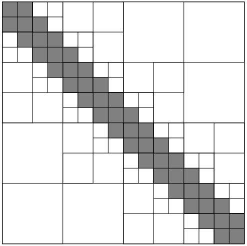
with $k_\varepsilon \ll b$ such that $\|B - XY^T\| \leq \varepsilon$

How to choose a good block partitioning of the matrix?



BLR matrix

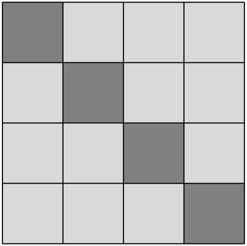
- Superlinear complexity
- Simple, flat structure



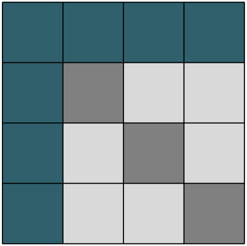
$\mathcal{H}$-matrix

- Nearly linear complexity
- Complex, hierarchical structure

BLR factorization: standard FCU variant

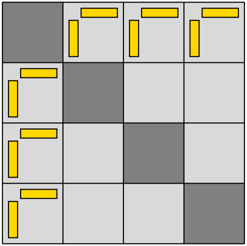


- FCU



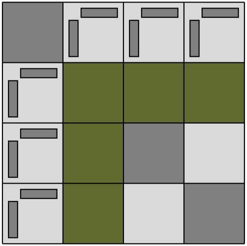
- FCU (Factor,
- Easy to handle numerical pivoting

BLR factorization: standard FCU variant



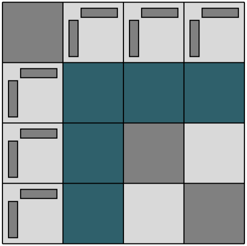
- FCU (Factor, Compress,
- Easy to handle numerical pivoting

BLR factorization: standard FCU variant



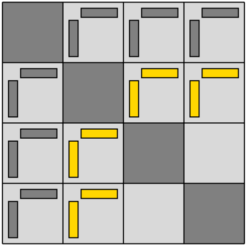
- FCU (Factor, Compress, Update)
- Easy to handle numerical pivoting

BLR factorization: standard FCU variant



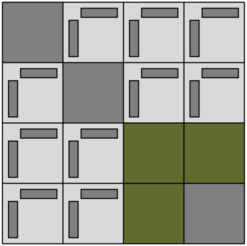
- FCU (Factor, Compress, Update)
- Easy to handle numerical pivoting

BLR factorization: standard FCU variant



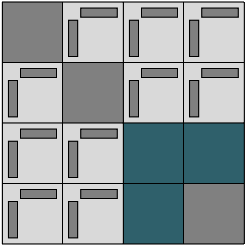
- FCU (Factor, Compress, Update)
- Easy to handle numerical pivoting

BLR factorization: standard FCU variant



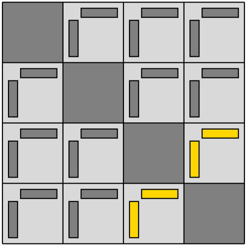
- FCU (Factor, Compress, Update)
- Easy to handle numerical pivoting

BLR factorization: standard FCU variant



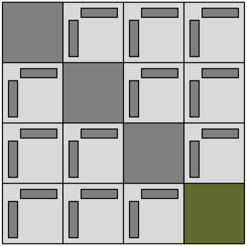
- FCU (Factor, Compress, Update)
- Easy to handle numerical pivoting

BLR factorization: standard FCU variant



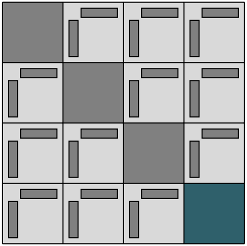
- FCU (Factor, Compress, Update)
- Easy to handle numerical pivoting

BLR factorization: standard FCU variant



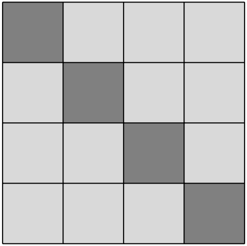
- FCU (Factor, Compress, Update)
- Easy to handle numerical pivoting

BLR factorization: standard FCU variant



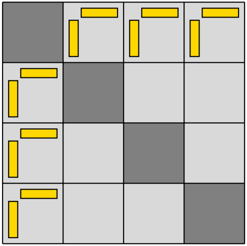
- FCU (Factor, Compress, Update)
- Easy to handle numerical pivoting

CFU factorization variant



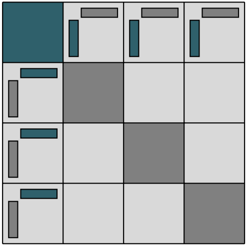
- CFU

CFU factorization variant



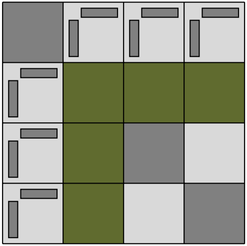
- CFU (Compress,

CFU factorization variant



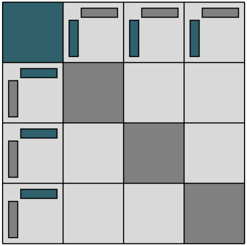
- CFU (Compress, Factor,
- Factor step is performed on compressed blocks $\Rightarrow$ reduced flops

CFU factorization variant



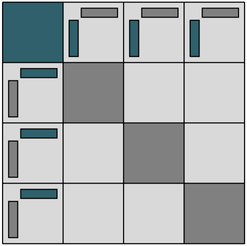
- CFU (Compress, Factor, Update)
- Factor step is performed on compressed blocks $\Rightarrow$ reduced flops

CFU factorization variant



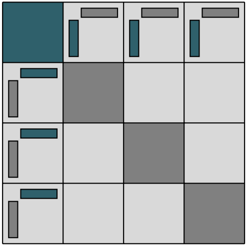
- CFU (Compress, Factor, Update)
- Factor step is performed on compressed blocks ⇒ reduced flops
- How can we handle numerical pivoting?

CFU factorization variant



- CFU (Compress, Factor, Update)
- Factor step is performed on compressed blocks $\Rightarrow$ reduced flops
- How can we handle numerical pivoting?
  - Restricting pivot choice to diagonal block is acceptable (in combination with a pivot delaying strategy)

CFU factorization variant



- CFU (Compress, Factor, Update)
- Factor step is performed on compressed blocks ⇒ reduced flops
- How can we handle numerical pivoting?
  - Restricting pivot choice to diagonal block is acceptable (in combination with a pivot delaying strategy)
  - Must still check entries in off-diagonal blocks: can be estimated from entries in low-rank blocks

Complexity of the BLR factorization

P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *On the Complexity of the Block Low-Rank Multifrontal Factorization*. SIAM J. Sci. Comput. (2017).

		storage	flops
dense	FR	$O(m^2)$	$O(m^3)$
	BLR	$O(m^{3/2})$	$O(m^2)$
	$\mathcal{H}$	$O(m \log m)$	$O(m \log^2 m)$

(assuming $r = O(1)$)

Complexity of the BLR factorization

P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *On the Complexity of the Block Low-Rank Multifrontal Factorization*. SIAM J. Sci. Comput. (2017).

		storage	flops
dense	FR	$O(m^2)$	$O(m^3)$
	BLR	$O(m^{3/2})$	$O(m^2)$
	$\mathcal{H}$	$O(m \log m)$	$O(m \log^2 m)$
sparse 2D	FR	$O(n \log n)$	$O(n^{3/2})$
	BLR	$O(n)$	$O(n \log n)$
	$\mathcal{H}$	$O(n)$	$O(n)$

(assuming $r = O(1)$)

- In a **2D** world hierarchical matrices would not be needed

Complexity of the BLR factorization

P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *On the Complexity of the Block Low-Rank Multifrontal Factorization*. SIAM J. Sci. Comput. (2017).

		storage	flops
dense	FR	$O(m^2)$	$O(m^3)$
	BLR	$O(m^{3/2})$	$O(m^2)$
	$\mathcal{H}$	$O(m \log m)$	$O(m \log^2 m)$
sparse 2D	FR	$O(n \log n)$	$O(n^{3/2})$
	BLR	$O(n)$	$O(n \log n)$
	$\mathcal{H}$	$O(n)$	$O(n)$
sparse 3D	FR	$O(n^{4/3})$	$O(n^2)$
	BLR	$O(n \log n)$	$O(n^{4/3})$
	$\mathcal{H}$	$O(n)$	$O(n)$

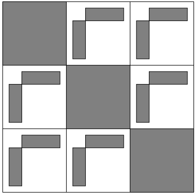
(assuming $r = O(1)$)

- In a **2D** world hierarchical matrices would not be needed
- Superlinear complexities in **3D**

Multilevel BLR format

Flop complexity (assuming $r = O(1)$):

	BLR	Hierar.
Dense	$O(m^2)$	$O(m \log^2 m)$
Sparse (2D)	$O(n \log n)$	$O(n)$
Sparse (3D)	$O(n^{1.33})$	$O(n)$



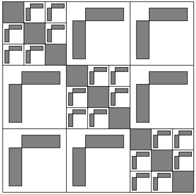
Multilevel BLR format

Flop complexity (assuming $r = O(1)$):

	BLR	Hierar.
Dense	$O(m^2)$	$O(m \log^2 m)$
Sparse (2D)	$O(n \log n)$	$O(n)$
Sparse (3D)	$O(n^{1.33})$	$O(n)$

Multilevel BLR (MBLR) format: refine full-rank blocks up to a constant number of levels ℓ

P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *Bridging the gap between flat and hierarchical low-rank matrix formats: the multilevel BLR format*. Submitted (2018).

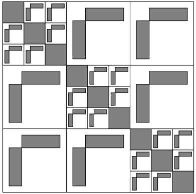


Multilevel BLR format

Flop complexity (assuming $r = O(1)$):

	$\ell = 1$	$\ell = 2$	Hierar.
Dense	$O(m^2)$	$O(m^{1.66})$	$O(m \log^2 m)$
Sparse (2D)	$O(n \log n)$	$O(n)$	$O(n)$
Sparse (3D)	$O(n^{1.33})$	$O(n^{1.11})$	$O(n)$

Multilevel BLR (MBLR) format: refine full-rank blocks up to a constant number of levels ℓ



P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *Bridging the gap between flat and hierarchical low-rank matrix formats: the multilevel BLR format*. Submitted (2018).

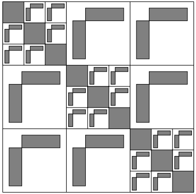
Multilevel BLR format

Flop complexity (assuming $r = O(1)$):

	$\ell = 1$	$\ell = 2$	$\ell = 3$	Hierar.
Dense	$O(m^2)$	$O(m^{1.66})$	$O(m^{1.5})$	$O(m \log^2 m)$
Sparse (2D)	$O(n \log n)$	$O(n)$	$O(n)$	$O(n)$
Sparse (3D)	$O(n^{1.33})$	$O(n^{1.11})$	$O(n \log n)$	$O(n)$

Multilevel BLR (MBLR) format: refine full-rank blocks up to a constant number of levels ℓ

P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *Bridging the gap between flat and hierarchical low-rank matrix formats: the multilevel BLR format*. Submitted (2018).

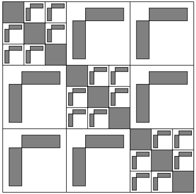


Multilevel BLR format

Flop complexity (assuming $r = O(1)$):

	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	Hierar.
Dense	$O(m^2)$	$O(m^{1.66})$	$O(m^{1.5})$	$O(m^{1.4})$	$O(m \log^2 m)$
Sparse (2D)	$O(n \log n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Sparse (3D)	$O(n^{1.33})$	$O(n^{1.11})$	$O(n \log n)$	$O(n)$	$O(n)$

Multilevel BLR (MBLR) format: refine full-rank blocks up to a constant number of levels ℓ



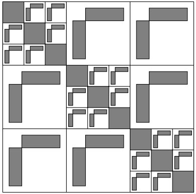
P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *Bridging the gap between flat and hierarchical low-rank matrix formats: the multilevel BLR format*. Submitted (2018).

Flop complexity (assuming $r = O(1)$):

	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	Hierar.
Dense	$O(m^2)$	$O(m^{1.66})$	$O(m^{1.5})$	$O(m^{1.4})$	$O(m \log^2 m)$
Sparse (2D)	$O(n \log n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Sparse (3D)	$O(n^{1.33})$	$O(n^{1.11})$	$O(n \log n)$	$O(n)$	$O(n)$

Multilevel BLR (MBLR) format: refine full-rank blocks up to a constant number of levels ℓ

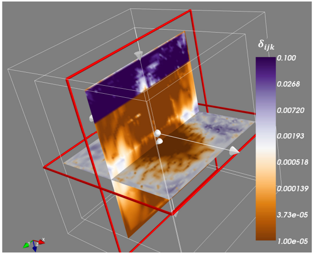
P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *Bridging the gap between flat and hierarchical low-rank matrix formats: the multilevel BLR format*. Submitted (2018).



With $r = O(1)$ only 4 levels are enough (even fewer needed for storage and sparse 2D complexities). With larger ranks more levels needed but **gain from adding more levels decreases rapidly**

Shared-memory performance analysis: an example

Matrix S3
Double complex (z) symmetric
Electromagnetics application (CSEM)
3.3 millions unknowns
Required accuracy: $\varepsilon = 10^{-7}$



D. Shantsev, P. Jaysaval, S. Kethulle de Ryhove, P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *Large-scale 3D EM modeling with a Block Low-Rank multifrontal direct solver*. Geophys. J. Int (2017).

	flops ($\times 10^{12}$)	time (1 core)	time (24 cores)
FR	78.0	7390	509
BLR	10.2	2242	307
ratio	7.7	3.3	1.7

7.7 gain in flops only translated to a **1.7** gain in time:
Can we do better?

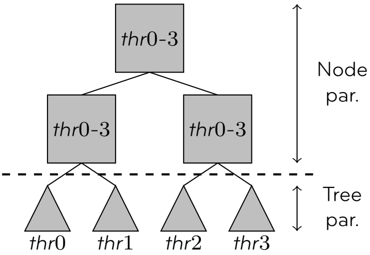
Improving the performance of BLR factorization

Variant name	time	FR/BLR ratio
Full-Rank	509	
BLR (FCU)	307	1.7

Improving the performance of BLR factorization

Tree parallelism improves performance by reducing the relative cost of the fronts at the bottom of the tree, which achieve poor compression

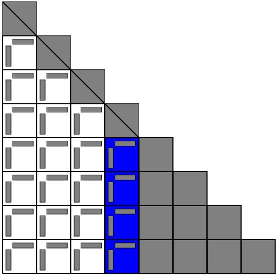
Variant name	time	FR/BLR ratio
Full-Rank	509	
+Tree par.	418	
BLR (FCU)	307	1.7
+Tree par.	221	1.9



Improving the performance of BLR factorization

Left-looking FCU improves performance thanks to a left-looking approach which reduces memory transfers

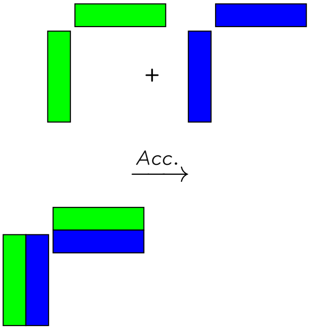
Variant name	time	FR/BLR ratio
Full-Rank	509	
+Tree par.	418	
BLR (FCU)	307	1.7
+Tree par.	221	1.9
+Left-looking	175	2.4



Improving the performance of BLR factorization

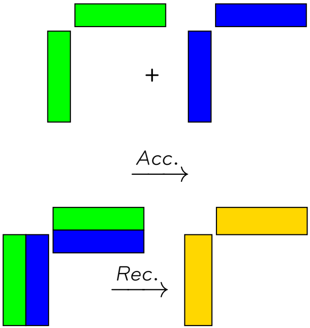
LUA improves performance because it accumulates multiple low-rank updates and applies them at once increasing the granularity of operations

Variant name	time	FR/BLR ratio
Full-Rank	509	
+Tree par.	418	
BLR (FCU)	307	1.7
+Tree par.	221	1.9
+Left-looking	175	2.4
+Accumulation	167	2.5



LUAR reduces complexity because recompresses accumulated updates on the fly

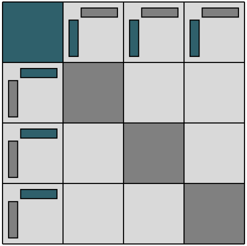
Variant name	time	FR/BLR ratio
Full-Rank	509	
+Tree par.	418	
BLR (FCU)	307	1.7
+Tree par.	221	1.9
+Left-looking	175	2.4
+Accumulation	167	2.5
+Recompression	160	2.6



Improving the performance of BLR factorization

CFU reduces complexity because solve operations are also done in low-rank

Variant name	time	FR/BLR ratio
Full-Rank	509	
+Tree par.	418	
BLR (FCU)	307	1.7
+Tree par.	221	1.9
+Left-looking	175	2.4
+Accumulation	167	2.5
+Recompression	160	2.6
+CFU	111	3.8



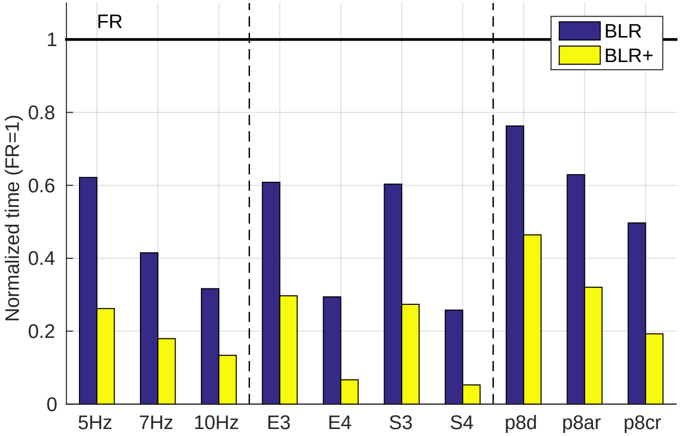
Improving the performance of BLR factorization

Variant name	time	FR/BLR ratio
Full-Rank	509	
+Tree par.	418	
BLR (FCU)	307	1.7
+Tree par.	221	1.9
+Left-looking	175	2.4
+Accumulation	167	2.5
+Recompression	160	2.6
+CFU	111	3.8

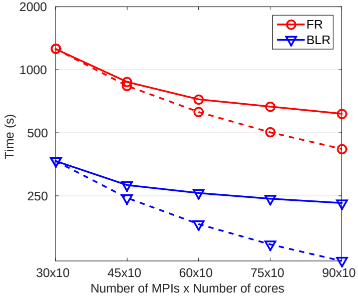
Converting the theoretical flop reduction into **actual time gains on modern architectures** requires careful algorithmic work

Results with the BLR MUMPS solver:

P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures*. ACM Trans. Math. Soft. (2018).



Results on $300 \rightarrow 900$ cores
(eos supercomputer, CALMIP)



Matrix 10Hz
Single complex (c) unsymmetric
Seismic imaging application (FWI)
17 millions unknowns
Required accuracy: $\varepsilon = 10^{-3}$

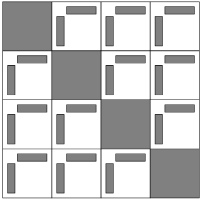
P. Amestoy, R. Brossier, A. Buttari, J.-Y. L'Excellent, T. Mary, L. Métivier, A. Miniussi, and S. Operto. *Fast 3D frequency-domain full waveform inversion with a parallel Block Low-Rank multifrontal direct solver: application to OBC data from the North Sea*. Geophysics (2016).

How to improve the scalability of the BLR factorization?
Two main difficulties:

- **Higher weight of communications:** flops reduced by **13** but volume of communications only by **2**
- **Unpredictability of compression:** more difficult to design good mapping and scheduling strategies

Rounding error analysis of BLR factorization

Why we need an error analysis



Each off-diagonal block B is approximated by a low-rank matrix $\widetilde{B}$ such that $\|B - \widetilde{B}\| \le \varepsilon \|B\|$
$\Rightarrow \|A - A_\varepsilon\| \le \varepsilon \|A\|$ with good norm choice
However:
$\|A - L_\varepsilon U_\varepsilon\| \ne \varepsilon$ because of rounding errors
$\Rightarrow$ **What is the overall accuracy $\|A - L_\varepsilon U_\varepsilon\|$?**

- Can we prove that $\|A - L_\varepsilon U_\varepsilon\| = O(\varepsilon)$? What is the role of the unit roundoff u?
- What is the error growth, i.e., how does the error depend on the matrix size n?
- How do the different variants (FCU, CFU, etc.) compare?
- Should we use an absolute threshold ($\|B - \widetilde{B}\| \le \varepsilon$) or a relative one ($\|B - \widetilde{B}\| \le \varepsilon \|B\|$)?

Main result: statement

Reminder

The full-rank LU factorization of $A \in \mathbb{R}^{n\times n}$ satisfies

$$\|A - LU\| \le nu\|L\|\|U\| + O(u^2)$$

Main result

The FCU BLR factorization of $A \in \mathbb{R}^{n\times n}$ with relative threshold ε satisfies

$$\|A - L_\varepsilon U_\varepsilon\| \le (nu + \varepsilon)\|L\|\|U\| + O(u\varepsilon) + O(u^2)$$

The proof is quite technical and based on *Stability of Block Algorithms with Fast Level-3 BLAS* (Demmel and Higham, 1992)

Main result: comments

Main result

The FCU BLR factorization of $A \in \mathbb{R}^{n\times n}$ with relative threshold ε satisfies

$$\|A - L_\varepsilon U_\varepsilon\| \le (nu + \varepsilon)\|L\|\|U\| + O(u\varepsilon) + O(u^2)$$

Main result: comments

Main result

The FCU BLR factorization of $A \in \mathbb{R}^{n\times n}$ with relative threshold ε satisfies

$$\|A - L_\varepsilon U_\varepsilon\| \le (nu + \varepsilon)\|L\|\|U\| + O(u\varepsilon) + O(u^2)$$

- $\|L\|\|U\| \le n^2 \rho_n \|A\|$ where ρ_n is the growth factor
  $\Rightarrow$ with partial pivoting, the BLR factorization is stable!

Main result: comments

Main result

The FCU BLR factorization of $A \in \mathbb{R}^{n\times n}$ with relative threshold ε satisfies

$$\|A - L_\varepsilon U_\varepsilon\| \le (nu + \varepsilon)\|L\|\|U\| + O(u\varepsilon) + O(u^2)$$

- $\|L\|\|U\| \le n^2 \rho_n \|A\|$ where ρ_n is the growth factor
  $\Rightarrow$ with partial pivoting, the BLR factorization is stable!
- Usually $\varepsilon \gg u$:

$\Rightarrow$ Role of u is limited

$\Rightarrow$ Very slow error growth

$\Rightarrow$ Usage of fast matrix arithmetic
may be stable in BLR

Main result: comments

Main result

The FCU BLR factorization of $A \in \mathbb{R}^{n\times n}$ with relative threshold ε satisfies

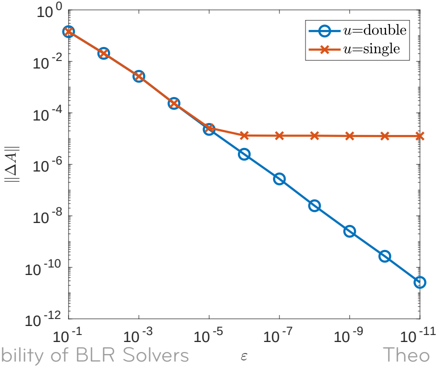
$$\|A - L_\varepsilon U_\varepsilon\| \le (nu + \varepsilon)\|L\|\|U\| + O(u\varepsilon) + O(u^2)$$

- $\|L\|\|U\| \le n^2 \rho_n \|A\|$ where ρ_n is the growth factor
  $\Rightarrow$ with partial pivoting, the BLR factorization is stable!
- Usually $\varepsilon \gg u$:

$\Rightarrow$ Role of u is limited

$\Rightarrow$ Very slow error growth

$\Rightarrow$ Usage of fast matrix arithmetic may be stable in BLR



Main result: comments

Main result

The FCU BLR factorization of $A \in \mathbb{R}^{n\times n}$ with relative threshold ε satisfies

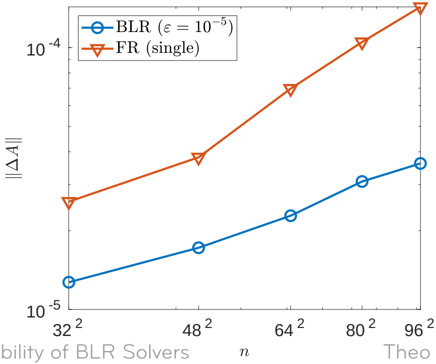
$$\|A - L_\varepsilon U_\varepsilon\| \le (nu + \varepsilon)\|L\|\|U\| + O(u\varepsilon) + O(u^2)$$

- $\|L\|\|U\| \le n^2 \rho_n \|A\|$ where ρ_n is the growth factor
  $\Rightarrow$ with partial pivoting, the BLR factorization is stable!
- Usually $\varepsilon \gg u$:

$\Rightarrow$ Role of u is limited

$\Rightarrow$ Very slow error growth

$\Rightarrow$ Usage of fast matrix arithmetic may be stable in BLR



Main result: comments

Main result

The FCU BLR factorization of $A \in \mathbb{R}^{n\times n}$ with relative threshold ε satisfies

$$\|A - L_\varepsilon U_\varepsilon\| \le (nu + \varepsilon)\|L\|\|U\| + O(u\varepsilon) + O(u^2)$$

- $\|L\|\|U\| \le n^2 \rho_n \|A\|$ where ρ_n is the growth factor
  $\Rightarrow$ with partial pivoting, the BLR factorization is stable!
- Usually $\varepsilon \gg u$:

$\Rightarrow$ Role of u is limited

$\Rightarrow$ Very slow error growth

$\Rightarrow$ Usage of fast matrix arithmetic may be stable in BLR

For example with Strassen's algorithm, $nu \to n^{\log_2 12} u \approx n^{3.6} u$

Ongoing work with C.-P. Jeannerod, C. Pernet, and D. Roche: *Exploiting fast matrix arithmetic within BLR factorizations*:
$O(n^2)$ complexity $\to O(n^{(\omega+1)/2})$
($\approx O(n^{1.9})$ for Strassen)

Relative vs absolute threshold

Theorem

The FCU BLR factorization of $A \in \mathbb{R}^{n\times n}$ with absolute threshold ε satisfies

$$\|A - L_\varepsilon U_\varepsilon\| \le (nu + \theta\varepsilon)\|L\|\|U\| + O(u\varepsilon) + O(u^2)$$

where $\theta = \sqrt{n/b - 1}\sum_{i=1}^{n/b} \|L_{ii}\| + \|U_{ii}\|$

The BLR factorization with absolute threshold

- ☹ Has a faster error growth
- ☹ Is scaling-dependent

Relative vs absolute threshold

Theorem

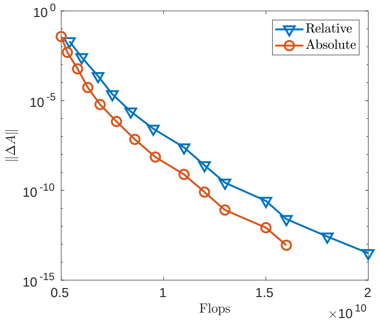
The FCU BLR factorization of $A \in \mathbb{R}^{n\times n}$ with absolute threshold ε satisfies

$$\|A - L_\varepsilon U_\varepsilon\| \le (nu + \theta\varepsilon)\|L\|\|U\| + O(u\varepsilon) + O(u^2)$$

where $\theta = \sqrt{n/b - 1}\sum_{i=1}^{n/b} \|L_{ii}\| + \|U_{ii}\|$

The BLR factorization with absolute threshold

- ☹ Has a faster error growth
- ☹ Is scaling-dependent
- ☺ Is more efficient in practice



Error analysis: CFU variant

Theorem

The CFU BLR factorization of $A \in \mathbb{R}^{n\times n}$ with relative threshold ε satisfies

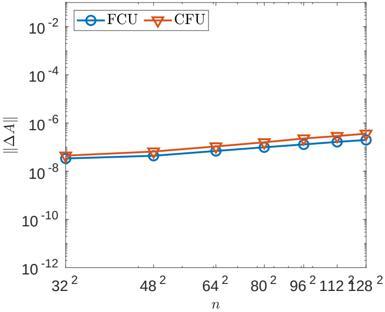
$$\|A - L_\varepsilon U_\varepsilon\| \le (nu + \varepsilon)\|L\|\|U\| + O(\kappa(A)u\varepsilon) + O(u^2)$$

Error analysis: CFU variant

Theorem

The CFU BLR factorization of $A \in \mathbb{R}^{n\times n}$ with relative threshold ε satisfies

$$\|A - L_\varepsilon U_\varepsilon\| \le (nu + \varepsilon)\|L\|\|U\| + O(\kappa(A)u\varepsilon) + O(u^2)$$

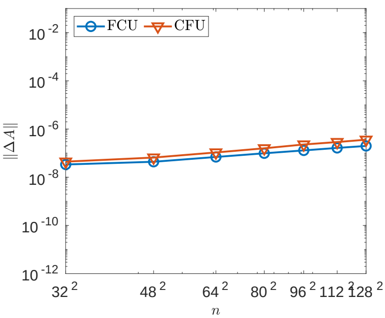


Error analysis: CFU variant

Theorem

The CFU BLR factorization of $A \in \mathbb{R}^{n \times n}$ with relative threshold ε satisfies

$$\|A - L_\varepsilon U_\varepsilon\| \le (nu + \varepsilon)\|L\|\|U\| + O(\kappa(A) u \varepsilon) + O(u^2)$$



Low-accuracy BLR preconditioners

Low-accuracy BLR preconditioners: storage

BLR factorization + GMRES solve with stopping tolerance 10^{-9}

Matrix	n	Time (s)		Storage (GB)	
		$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-8}$	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-8}$
audikw_1	1.0M	1163	69	5	10
Bump_2911	2.9M	–	282	34	56
Emilia_923	0.9M	304	63	7	12
Fault_639	0.6M	–	45	5	9
Ga41As41H72	0.3M	–	76	12	17
Hook_1498	1.5M	902	75	6	11
Si87H76	0.2M	–	62	10	14

Low-accuracy BLR solvers:

☹ are slower and less robust

☺ but require much less storage

Improved preconditioner: context

Objective

- Compute solution to linear system $Ax = b$
- $A \in \mathbb{R}^{n\times n}$ is ill conditioned

LU-based preconditioner

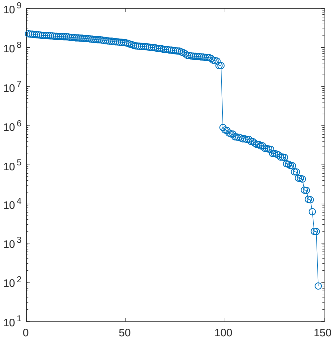
1. Compute approximate factorization $A = \widehat{L}\widehat{U} + \Delta A$
   - Half-precision factorization
   - Incomplete LU factorization
   - Structured matrix factorization: Block Low-Rank, $\mathcal{H}$, HSS,...
2. Solve $\Pi_{LU}Ax = \Pi_{LU}b$ with $\Pi_{LU} = \widehat{U}^{-1}\widehat{L}^{-1}$ via some iterative method

- Convergence to solution may be slow or fail

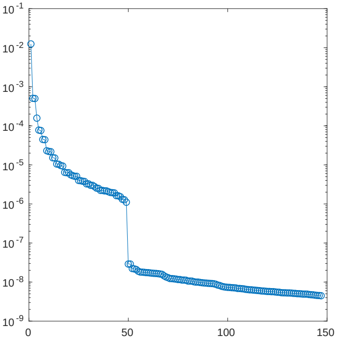
⇒ **Objective: accelerate convergence**

Improved preconditioner: key observation

Matrix lund_a ($n = 147$, $\kappa(A) = 2.8e{+}06$)



SVD of A



SVD of A^{-1}

- Often, A is ill conditioned due to a small number of small singular values
- Then, A^{-1} is numerically low-rank

Improved preconditioner: key idea

Factorization error might be low-rank?

Let the error $E = \widehat{U}^{-1}\widehat{L}^{-1}A - I = \widehat{U}^{-1}\widehat{L}^{-1}(\widehat{L}\widehat{U} + \Delta A) - I$
$$= \widehat{U}^{-1}\widehat{L}^{-1}\Delta A \approx A^{-1}\Delta A$$

Does E retain the low-rank property of A^{-1}?

A novel preconditioner

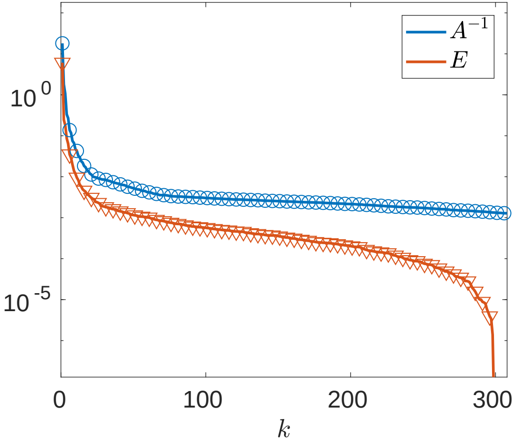
Consider the preconditioner

$$\Pi_{E_k} = (I + E_k)^{-1}\Pi_{LU}$$

with E_k a rank-k approximation to E.

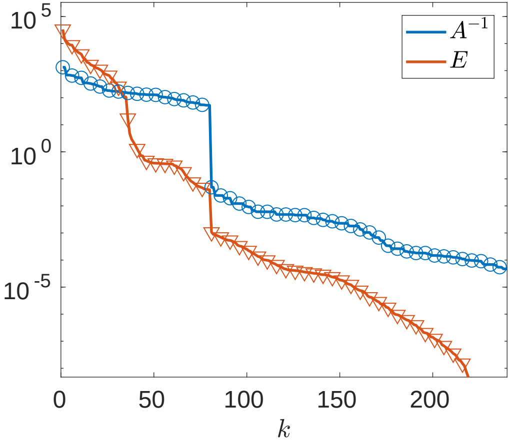
- If $E = E_k$, $\Pi_{E_k} = A^{-1}$
- If $E \approx E_k$ for some small k, Π_{E_k} can be computed cheaply

Typical SV distributions of A^{-1} and E



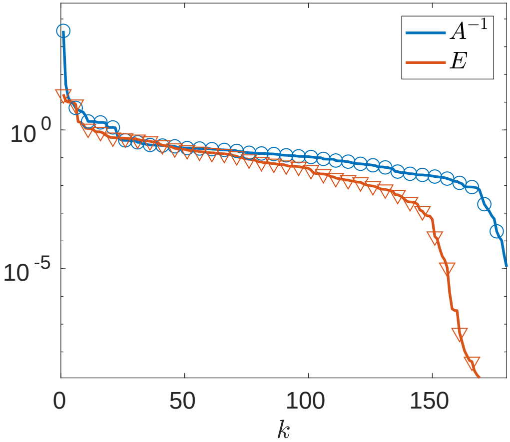
Matrix cz308

Typical SV distributions of A^{-1} and E



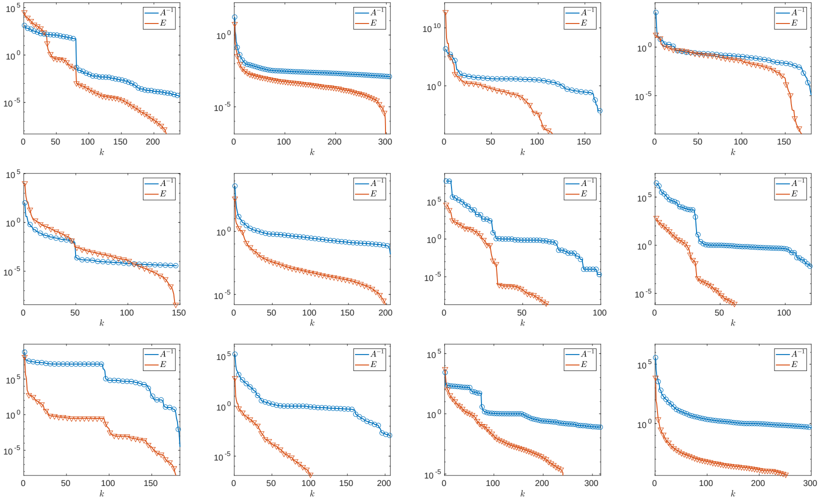
Matrix steam1

Typical SV distributions of A^{-1} and E



Matrix rajat14

Typical SV distributions of A^{-1} and E



We did **not** specifically select matrices for which A^{-1} is low-rank!

Computing E_k

We need to compute a rank-k approximation of

$$E = \widehat{U}^{-1}\widehat{L}^{-1}A - I$$

E cannot be built explicitly! $\Rightarrow$ use **randomized** method

Algorithm 1 Randomized SVD via direct SVD of $V^T E$.

1: Sample E: $S = E\Omega$, with Ω a $n \times (k+p)$ random matrix.
2: Orthonormalize S: $V = \text{qr}(S)$. $\{\Rightarrow E \approx VV^T E.\}$
3: Compute truncated SVD $V^T E \approx X_k \Sigma_k Y_k^T$.
4: $E_k \approx (VX_k)\Sigma_k Y_k^T$.

Results for $\varepsilon = 10^{-2}$:

Matrix	Π_{LU}		Π_{E_k}	
	Iter.	Time	Iter.	Time
audikw_1	691	1163	331	625
Bump_2911	–	–	284	1708
Emilia_923	174	304	136	267
Fault_639	–	–	294	345
Ga41As41H72	–	–	135	143
Hook_1498	417	902	356	808
Si87H76	–	–	131	116

$\Rightarrow$ **performance and robustness improvement with zero storage overhead**

Probabilistic rounding error analysis

Context and motivation

Floating-point arithmetic model

$$\mathrm{fl}(a \text{ op } b) = (a \text{ op } b)(1+\delta), \quad |\delta| \le u, \quad \text{op} \in \{+,-,\times,/\}$$

	fp64 (double)	fp32 (single)	fp16 (half)	fp8 (quarter)
u	$2^{-53} \approx 10^{-16}$	$2^{-24} \approx 10^{-8}$	$2^{-11} \approx 10^{-4}$	$2^{-4} \approx 10^{-2}$

- In many numerical linear algebra computations, traditional error bounds are proportional to nu, e.g., for LU factorization:

$$|A - LU| \le nu|L||U|$$

$\Rightarrow$ No guarantees if nu is large: issue of growing importance with the rise of large-scale, mixed-precision computations

Context and motivation

Floating-point arithmetic model

$$\mathrm{fl}(a \text{ op } b) = (a \text{ op } b)(1+\delta), \quad |\delta| \le u, \quad \text{op} \in \{+,-,\times,/\}$$

	fp64 (double)	fp32 (single)	fp16 (half)	fp8 (quarter)
u	2^{-53} $\approx 10^{-16}$	2^{-24} $\approx 10^{-8}$	2^{-11} $\approx 10^{-4}$	2^{-4} $\approx 10^{-2}$

- In many numerical linear algebra computations, traditional error bounds are proportional to nu, e.g., for LU factorization:

$$|A - LU| \le nu|L||U|$$

$\Rightarrow$ No guarantees if nu is large: issue of growing importance with the rise of large-scale, mixed-precision computations

- This issue is independent of low-rank solvers, but...
  - Improved asymptotic complexity $\Rightarrow$ larger n
  - Error bound dominated by ε $\Rightarrow$ larger u

$\Rightarrow$ $nu > 1$ **will happen fast with low-rank solvers**

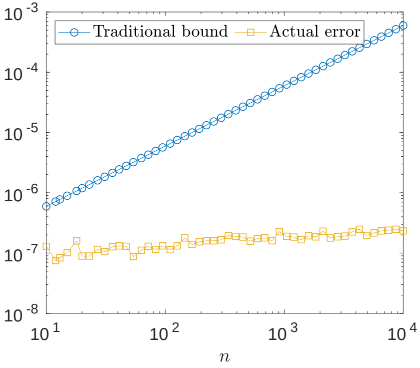
Traditional bounds are pessimistic

The issue is that traditional bounds are worst-case bounds, and are thus pessimistic on average

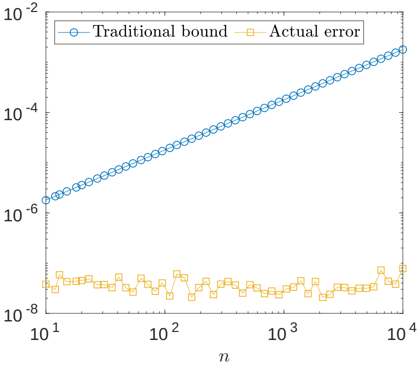
Traditional bounds are pessimistic

The issue is that traditional bounds are worst-case bounds, and are thus pessimistic on average

Matrix–vector product (fp32)

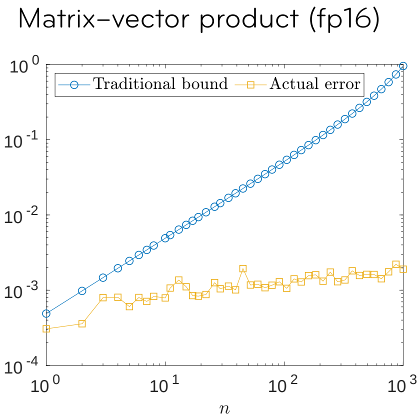


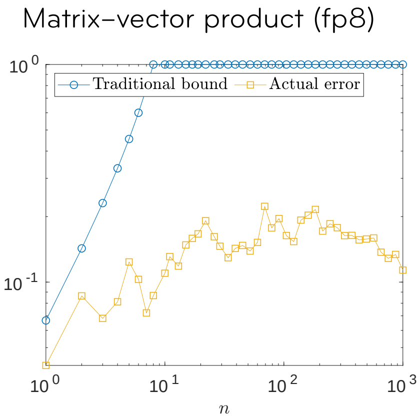
Solution of $Ax = b$ (fp32)



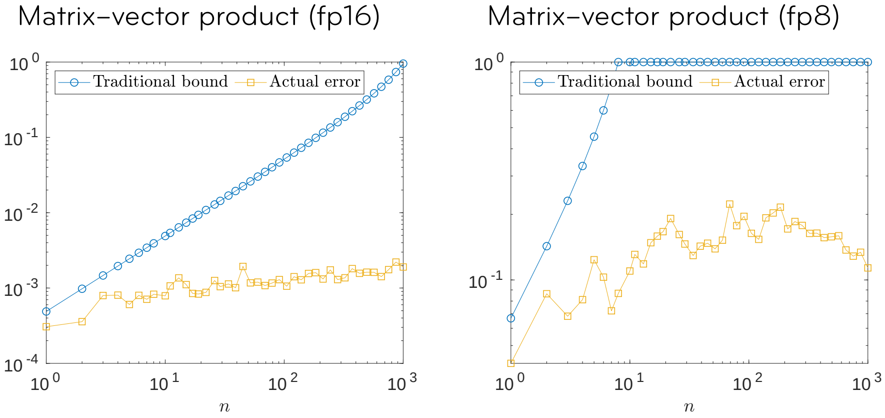
Traditional bounds are pessimistic

The issue is that traditional bounds are worst-case bounds, and are thus pessimistic on average





The issue is that traditional bounds are worst-case bounds, and are thus pessimistic on average



⇒ Traditional bounds do not provide a realistic picture of the typical behavior of numerical computations

Key intuition

- Consider the accumulated effect of n rounding errors

$$s = \sum_{i=1}^{n} \delta_i$$

- The worst-case bound $|s| \le nu$ is attained when all δ_i have identical sign and maximal magnitude, which intuitively seems very unlikely
- Assume δ_i are random independent variables of mean zero
- Then, the central limit theorem states that if n is sufficiently large, then

$$s/\sqrt{n} \sim \mathcal{N}(0, u)$$

$\Rightarrow$ $|s| \le \lambda\sqrt{n}u$, with λ a small constant, holds with high probability (e.g., 99.7% with $\lambda = 3$ by the 3-sigma rule)

The rule of thumb

This probabilistic approach had led to the following rule of thumb

> *In general, the statistical distribution of the rounding errors will reduce considerably the function of n occurring in the relative errors. We might expect in each case that this function should be replaced by something which is no bigger than its square root.*
>
> — *James Wilkinson, 1961*

However, no rigorous result along these lines exists for a wide class of algorithms

The rule of thumb

This probabilistic approach had led to the following rule of thumb

> *In general, the statistical distribution of the rounding errors will reduce considerably the function of n occurring in the relative errors. We might expect in each case that this function should be replaced by something which is no bigger than its square root.*
>
> — *James Wilkinson, 1961*

However, no rigorous result along these lines exists for a wide class of algorithms

Our contribution:
We provide the first rigorous foundation for this rule of thumb
by computing rigorous error bounds
that hold with probability at least a certain value
for a wide class of linear algebra algorithms

Objective and assumptions

Fundamental lemma in backward error analysis

If $|\delta_i| \le u$ for $i = 1 : n$ and $nu < 1$, then

$$\prod_{i=1}^{n}(1+\delta_i) = 1 + \theta_n, \quad |\theta_n| \le \gamma_n \le nu + O(u^2)$$

Objective and assumptions

Fundamental lemma in backward error analysis

If $|\delta_i| \le u$ for $i = 1 : n$ and $nu < 1$, then

$$\prod_{i=1}^{n}(1+\delta_i) = 1 + \theta_n, \quad |\theta_n| \le \gamma_n \le nu + O(u^2)$$

We seek an anologous result by using the following model

Probabilistic model of rounding errors

In the computation of interest, the quantities δ in the model

$$\mathrm{fl}(a \text{ op } b) = (a \text{ op } b)(1+\delta), \quad |\delta| \le u, \quad \text{op} \in \{+, -, \times, /\}$$

associated with every pair of operands are independent random variables of mean zero.

There is no claim that ordinary rounding and chopping are random processes, or that successive errors are independent. ***The question to be decided is whether or not these particular probabilistic models of the processes will adequately describe what actually happens.***

— Hull and Swenson, 1966

Proof sketch

First step: transform the product in a sum by taking the logarithm

$$S = \log \prod_{i=1}^{n} (1 + \delta_i) = \sum_{i=1}^{n} \log(1 + \delta_i)$$

Proof sketch

First step: transform the product in a sum by taking the logarithm

$$S = \log \prod_{i=1}^{n} (1+\delta_i) = \sum_{i=1}^{n} \log(1+\delta_i)$$

Second step: apply Hoeffding's concentration inequality:

Hoeffding's inequality

Let X_1, ..., X_n be random independent variables satisfying $|X_i| \le c_i$. Then the sum $S = \sum_{i=1}^{n} X_i$ satisfies

$$\Pr(|S - \mathbb{E}(S)| \ge \xi) \le 2 \exp\left(-\frac{\xi^2}{2\sum_{i=1}^{n} c_i^2}\right)$$

to $X_i = \log(1+\delta_i) \Rightarrow$ requires bounding $\log(1+\delta_i)$ and $\mathbb{E}(\log(1+\delta_i))$ using Taylor expansions

Proof sketch

First step: transform the product in a sum by taking the logarithm

$$S = \log \prod_{i=1}^{n} (1 + \delta_i) = \sum_{i=1}^{n} \log(1 + \delta_i)$$

Second step: apply Hoeffding's concentration inequality:

Hoeffding's inequality

Let X_1, ..., X_n be random independent variables satisfying $|X_i| \le c_i$. Then the sum $S = \sum_{i=1}^{n} X_i$ satisfies

$$\Pr(|S - \mathbb{E}(S)| \ge \xi) \le 2 \exp\left(-\frac{\xi^2}{2 \sum_{i=1}^{n} c_i^2}\right)$$

to $X_i = \log(1 + \delta_i)$ $\Rightarrow$ requires bounding $\log(1 + \delta_i)$ and $\mathbb{E}\left(\log(1 + \delta_i)\right)$ using Taylor expansions

Third step: retrieve the result by taking the exponential of S

Our main result

Main result

Let δ_i, $i = 1 : n$, be independent random variables of mean zero such that $|\delta_i| \le u$. Then, for any constant $\lambda > 0$, the relation

$$\prod_{i=1}^{n}(1+\delta_i) = 1 + \theta_n, \quad |\theta_n| \le \widetilde{\gamma}_n(\lambda) := \exp\left(\lambda\sqrt{n}u + \frac{nu^2}{1-u}\right) - 1$$

$$\le \lambda\sqrt{n}u + O(u^2)$$

holds with probability of failure $P(\lambda) = 2\exp\left(-\lambda^2(1-u)^2/2\right)$

Our main result

Main result

Let δ_i, $i = 1:n$, be independent random variables of mean zero such that $|\delta_i| \le u$. Then, for any constant $\lambda > 0$, the relation

$$\prod_{i=1}^{n}(1+\delta_i) = 1+\theta_n, \quad |\theta_n| \le \widetilde{\gamma}_n(\lambda) := \exp\left(\lambda\sqrt{n}u + \frac{nu^2}{1-u}\right) - 1$$
$$\le \lambda\sqrt{n}u + O(u^2)$$

holds with probability of failure $P(\lambda) = 2\exp\left(-\lambda^2(1-u)^2/2\right)$

Key features:

- Exact bound, not first order
- $nu < 1$ not required
- No "n is sufficiently large" assumption (achieved by replacing the central limit theorem by Hoeffding's inequality)
- Small values of λ suffice: $P(1) \approx 0.27$, $P(5) \le 10^{-5}$

Application to numerical linear algebra

Bounds for many numerical linear algebra algorithms are obtained by the repeated application of our main result. For example:

Probabilistic bound for LU factorization

Let $LU = A + \Delta A$ be the LU factors computed by Gaussian elimination of $A \in \mathbb{R}^{n\times n}$. Then, for any constant $\lambda > 0$, the relation

$$|\Delta A| \le \widetilde{\gamma}_n(\lambda)|L||U|, \quad |\widetilde{\gamma}_n(\lambda)| \le \lambda\sqrt{n}u + O(u^2)$$

holds with probability of failure $(n^3/3 + n^2/2 + 7n/6)P(\lambda)$

Application to numerical linear algebra

Bounds for many numerical linear algebra algorithms are obtained by the repeated application of our main result. For example:

Probabilistic bound for LU factorization

Let $LU = A + \Delta A$ be the LU factors computed by Gaussian elimination of $A \in \mathbb{R}^{n \times n}$. Then, for any constant $\lambda > 0$, the relation

$$|\Delta A| \le \widetilde{\gamma}_n(\lambda)|L||U|, \quad |\widetilde{\gamma}_n(\lambda)| \le \lambda\sqrt{n}u + O(u^2)$$

holds with probability of failure $(n^3/3 + n^2/2 + 7n/6)P(\lambda)$

We wish to keep the probabilities independent of n! Fortunately:

$$O(n^3)P(\lambda) = O(1) \quad \Rightarrow \quad \lambda = O(\sqrt{\log n})$$

$\Rightarrow$ error grows no faster than $\sqrt{n \log n}u$

Application to numerical linear algebra

Bounds for many numerical linear algebra algorithms are obtained by the repeated application of our main result. For example:

Probabilistic bound for LU factorization

Let $LU = A + \Delta A$ be the LU factors computed by Gaussian elimination of $A \in \mathbb{R}^{n \times n}$. Then, for any constant $\lambda > 0$, the relation

$$|\Delta A| \le \widetilde{\gamma}_n(\lambda)|L||U|, \quad |\widetilde{\gamma}_n(\lambda)| \le \lambda\sqrt{n}u + O(u^2)$$

holds with probability of failure $(n^3/3 + n^2/2 + 7n/6)P(\lambda)$

We wish to keep the probabilities independent of n! Fortunately:

$$O(n^3)P(\lambda) = O(1) \quad \Rightarrow \quad \lambda = O(\sqrt{\log n})$$

$\Rightarrow$ error grows no faster than $\sqrt{n \log n}u$

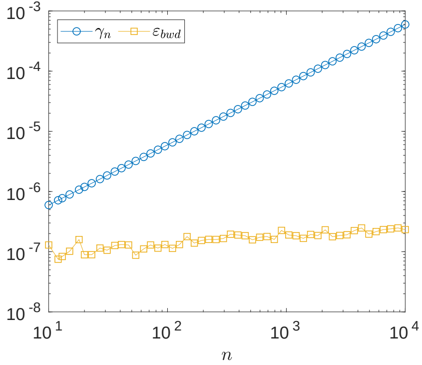
Moreover the constant hidden in the big O is small:

$$P(13) \le 10^{-5} \text{ for } n \le 10^{10}$$

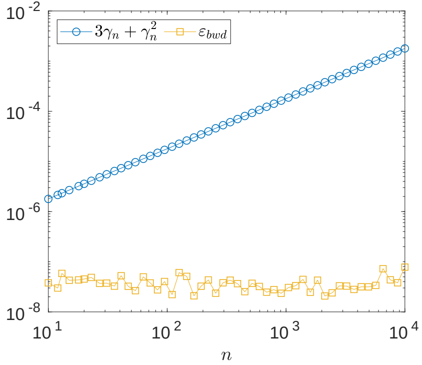
Experimental setting

- We use MATLAB R2018b and set `rng(1)` for reproducibility
- fp16 and fp8 are simulated with the rounding function `chop.m` from the Matrix Computation Toolbox
- We use both random matrices with entries drawn from the uniform $[-1,1]$ or $[0,1]$ distribution and real-life matrices from the SuiteSparse collection
- We compare the bounds γ_n and $\widetilde{\gamma}_n(\lambda)$ with the componentwise backward error ε_{bwd} measured as (Oettli–Prager):
  - Matrix–vector product $y = Ax$: $\varepsilon_{bwd} = \max_i \frac{|\widehat{y}-y|_i}{(|A||x|)_i}$
  - Solution to $Ax = b$ via LU factorization: $\varepsilon_{bwd} = \max_i \frac{|A\widehat{x}-b|_i}{(|\widehat{L}||\widehat{U}||\widehat{x}|)_i}$
- Our codes are available online: https://gitlab.com/theo.andreas.mary/proberranalysis

Matrix–vector product (fp32)

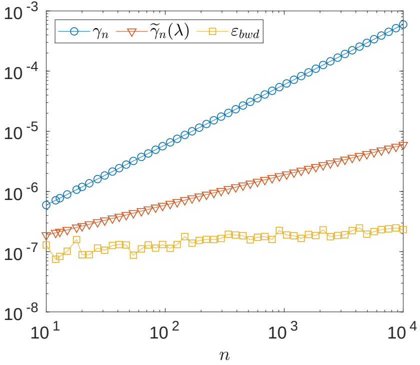


Solution of $Ax = b$ (fp32)

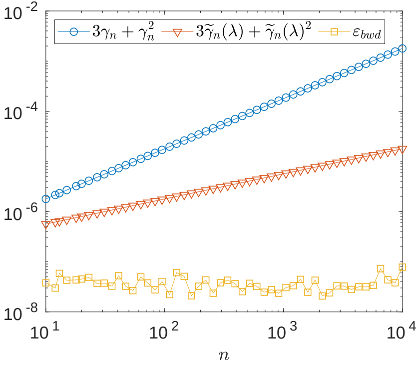


Experimental results with $[-1, 1]$ entries

Matrix–vector product (fp32)



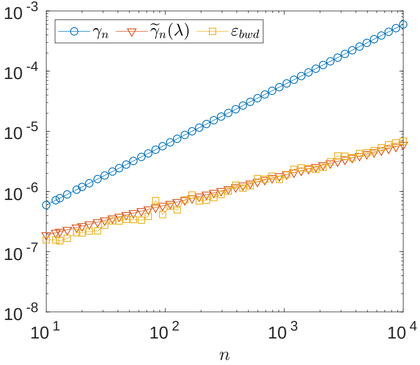
Solution of $Ax = b$ (fp32)



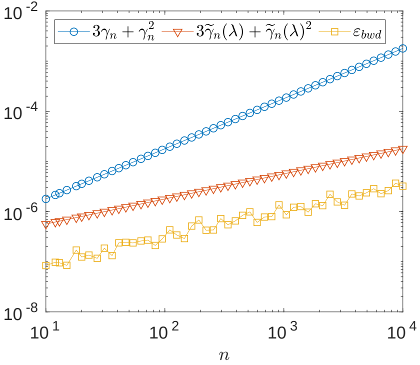
- The probabilistic bound is much closer to the actual error
- However for $[-1, 1]$ entries it is still pessimistic

Experimental results with $[0,1]$ entries

Matrix–vector product (fp32)



Solution of $Ax = b$ (fp32)

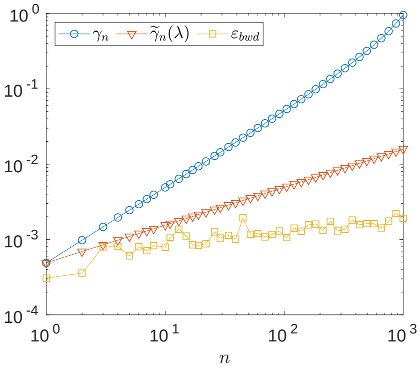


- Probabilistic bound is plotted with $\lambda = 1 \Rightarrow P(\lambda)$ is pessimistic...
- ...but $\widetilde{\gamma}_n$ bound itself can be sharp and successfully captures the $\sqrt{n}$ error growth

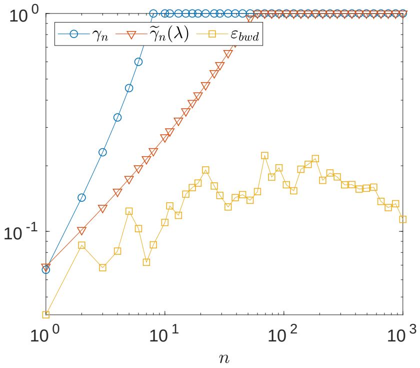
$\Rightarrow$ Therefore the bounds cannot be further improved without further assumptions

Experimental results with low precisions ($[-1, 1]$ entries)

Matrix–vector product (fp16)

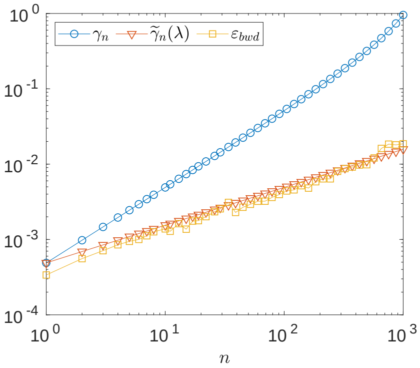


Matrix–vector product (fp8)

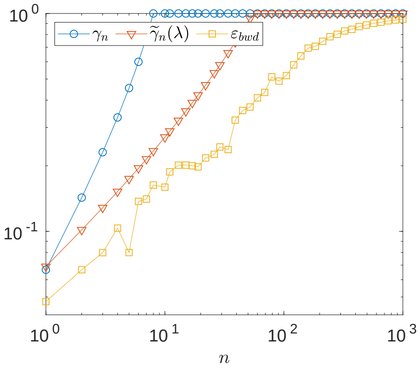


- Importance of the probabilistic bound becomes even clearer for lower precisions

Matrix–vector product (fp16)

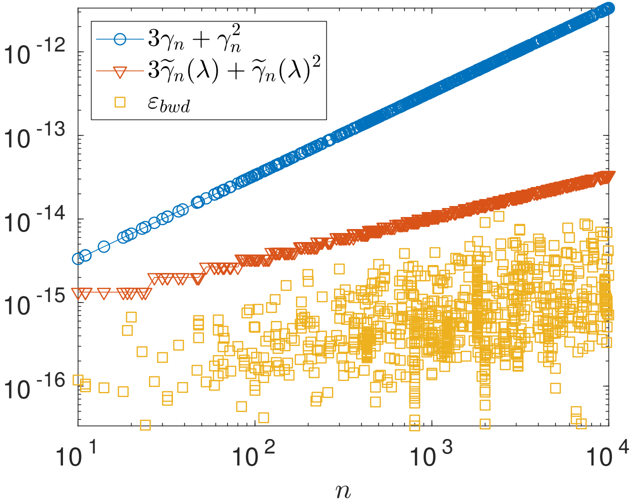


Matrix–vector product (fp8)



- Importance of the probabilistic bound becomes even clearer for lower precisions

Solution of $Ax = b$ (fp64),
for 943 matrices from the SuiteSparse collection



An example where rounding errors are not independent

Inner product of two constant vectors:

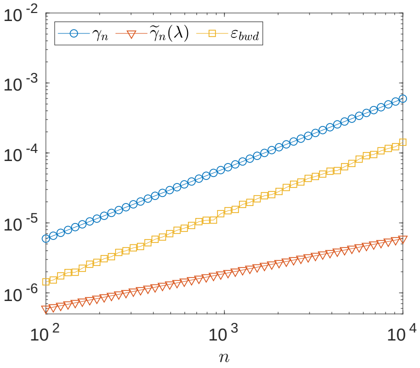
$$\begin{aligned} & s_{i+1} = s_i + a_i b_i = s_i + c \\ \Rightarrow \quad & \widehat{s}_{i+1} = (\widehat{s}_i + c)(1 + \delta_i) \end{aligned}$$

An example where rounding errors are not independent

Inner product of two constant vectors:

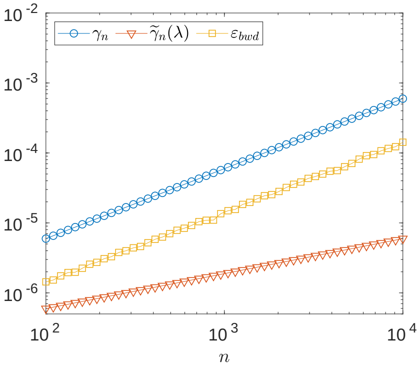
$$s_{i+1} = s_i + a_i b_i = s_i + c$$

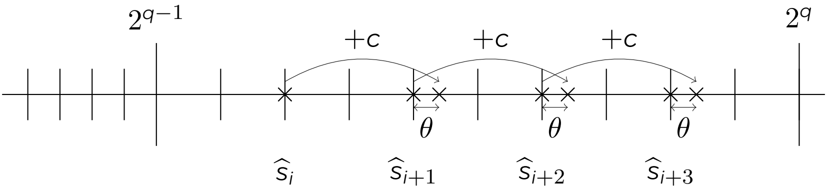
$$\Rightarrow \quad \widehat{s}_{i+1} = (\widehat{s}_i + c)(1 + \delta_i)$$



Inner product of two constant vectors:

$$s_{i+1} = s_i + a_i b_i = s_i + c$$
$$\Rightarrow \quad \widehat{s}_{i+1} = (\widehat{s}_i + c)(1 + \delta_i)$$



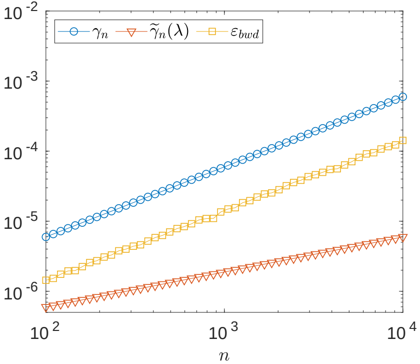


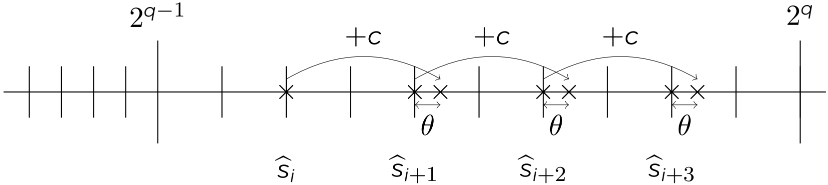
An example where rounding errors are not independent

Inner product of two constant vectors:

$$s_{i+1} = s_i + a_i b_i = s_i + c$$
$$\Rightarrow \quad \widehat{s}_{i+1} = (\widehat{s}_i + c)(1 + \delta_i)$$

$\Rightarrow$ $\delta_i = \theta$ is constant within intervals $[2^{q-1}; 2^q]$





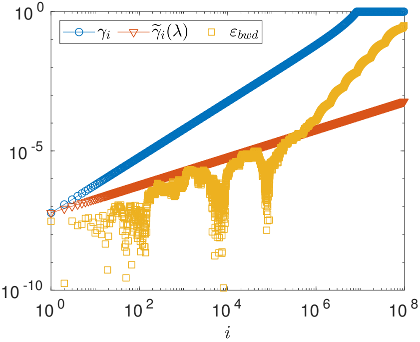
An example where rounding errors have nonzero mean

Inner product of two very large nonnegative vectors:

$$s_{i+1} = s_i + a_i b_i \quad \Rightarrow \quad \widehat{s}_{i+1} = (\widehat{s}_i + a_i b_i)(1 + \delta_i)$$

Inner product of two very large nonnegative vectors:

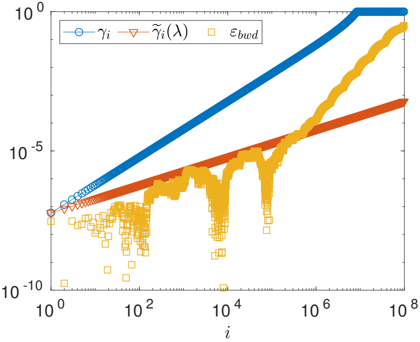
$$s_{i+1} = s_i + a_i b_i \quad \Rightarrow \quad \widehat{s}_{i+1} = (\widehat{s}_i + a_i b_i)(1+\delta_i)$$



An example where rounding errors have nonzero mean

Inner product of two very large nonnegative vectors:

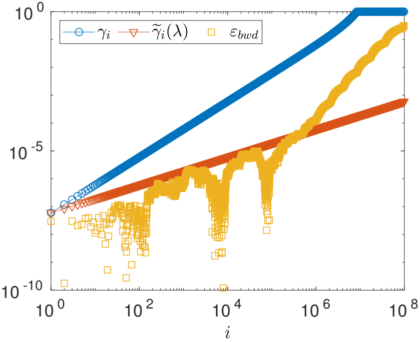
$$s_{i+1} = s_i + a_i b_i \quad \Rightarrow \quad \widehat{s}_{i+1} = (\widehat{s}_i + a_i b_i)(1 + \delta_i)$$



Explanation: s_i keeps increasing, at some point, it becomes so large that $\widehat{s}_{i+1} = \widehat{s}_i \Rightarrow \delta_i = -a_i b_i / (\widehat{s}_i + a_i b_i) < 0$

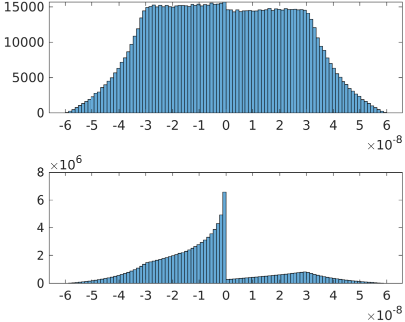
Inner product of two very large nonnegative vectors:

$$s_{i+1} = s_i + a_i b_i \quad \Rightarrow \quad \widehat{s}_{i+1} = (\widehat{s}_i + a_i b_i)(1 + \delta_i)$$



Top: $1 \leq n \leq 10^6$

Bottom: $10^6 \leq n \leq 10^8$



Explanation: s_i keeps increasing, at some point, it becomes so large that $\widehat{s}_{i+1} = \widehat{s}_i \Rightarrow \delta_i = -a_i b_i / (\widehat{s}_i + a_i b_i) < 0$

Conclusion

Conclusion

Takeaway messages

BLR solvers are numerically stable (with numerical pivoting) and can efficiently exploit low-precision floating-point arithmetic when used as low-accuracy preconditioners

Perspectives

- Rounding error analysis of multilevel and hierarchical solvers
- Probabilistic error analysis of low-rank factorizations
- Exploiting half precision within low-rank preconditioners
- Error analysis of low-rank preconditioners with iterative refinement

Slides and papers available here

`bit.ly/theomary` (list of references on next slide)

References

P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *On the Complexity of the Block Low-Rank Multifrontal Factorization*. SIAM J. Sci. Comput. (2017).

P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *Bridging the gap between flat and hierarchical low-rank matrix formats: the multilevel BLR format*. Submitted (2018).

P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures*. ACM Trans. Math. Soft. (2018).

C. Gorman, G. Chavez, P .Ghysels, T. Mary, F.-H. Rouet, and X. S. Li. *Matrix-free Construction of HSS Representation Using Adaptive Randomized Sampling*. Submitted (2018).

N. Higham and T. Mary. *A New Preconditioner that Exploits Low-Rank Approximations to Factorization Error*. SIAM J. Sci. Comp (2018).

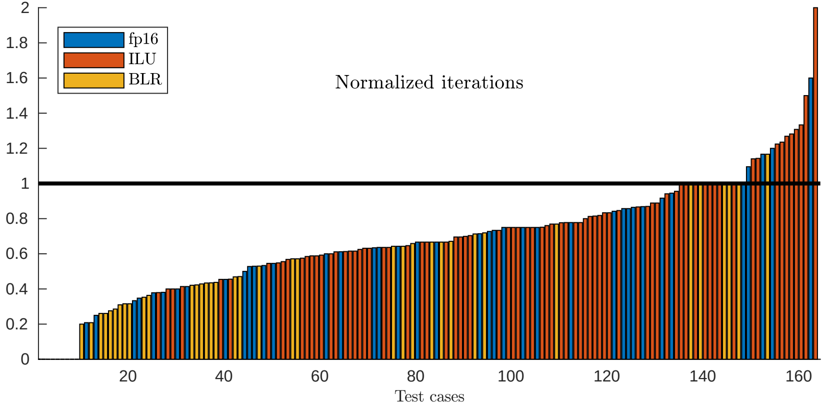
N. Higham and T. Mary. *A New Approach to Probabilistic Rounding Error Analysis*. Submitted (2018).

P. Amestoy, R. Brossier, A. Buttari, J.-Y. L'Excellent, T. Mary, L. Métivier, A. Miniussi, and S. Operto. *Fast 3D frequency-domain full waveform inversion with a parallel Block Low-Rank multifrontal direct solver: application to OBC data from the North Sea*. Geophysics (2016).

D. Shantsev, P. Jaysaval, S. Kethulle de Ryhove, P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. *Large-scale 3D EM modeling with a Block Low-Rank multifrontal direct solver*. Geophys. J. Int (2017).

Backup slides

Black-box setting: use $p = 10$ and $k =$ num. rank at acc. 10^{-7}



Storage overhead: formula

We need to store E_k: two dense $n \times k$ matrices

$\Rightarrow$ but only needed after factorization

Storage overhead: formula

We need to store E_k: two dense $n \times k$ matrices
$\Rightarrow$ but only needed after factorization

Traditional multifrontal storage is $S_A + S_{LU} + S_{CB}$

- S_A = storage for matrix A
- S_{LU} = storage for (BLR) LU factors
- S_{CB} = storage for contribution blocks $\Rightarrow$ temporary storage during factorization

Storage overhead: formula

We need to store E_k: two dense $n \times k$ matrices
$\Rightarrow$ but only needed after factorization

Traditional multifrontal storage is $S_A + S_{LU} + S_{CB}$

- S_A = storage for matrix A
- S_{LU} = storage for (BLR) LU factors
- S_{CB} = storage for contribution blocks $\Rightarrow$ temporary storage during factorization

Thus, S_{CB} and S_{E_k} do not overlap!

- Factorization storage: $S_A + S_{LU} + S_{CB}$
- Solution storage: $S_A + S_{LU} + S_{E_k}$

$\Rightarrow$ Total storage: $S_A + S_{LU} + \max(S_{CB}, S_{E_k})$

Storage overhead: formula

We need to store E_k: two dense $n \times k$ matrices
$\Rightarrow$ but only needed after factorization

Traditional multifrontal storage is $S_A + S_{LU} + S_{CB}$

- S_A = storage for matrix A
- S_{LU} = storage for (BLR) LU factors
- S_{CB} = storage for contribution blocks $\Rightarrow$ temporary storage during factorization

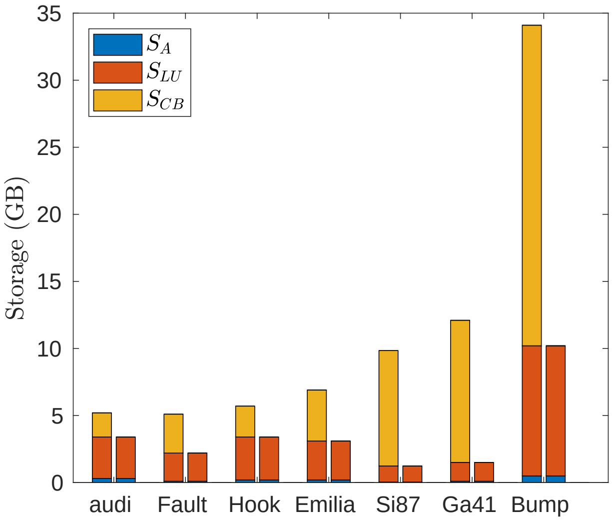
Thus, S_{CB} and S_{E_k} do not overlap!

- Factorization storage: $S_A + S_{LU} + S_{CB}$
- Solution storage: $S_A + S_{LU} + S_{E_k}$

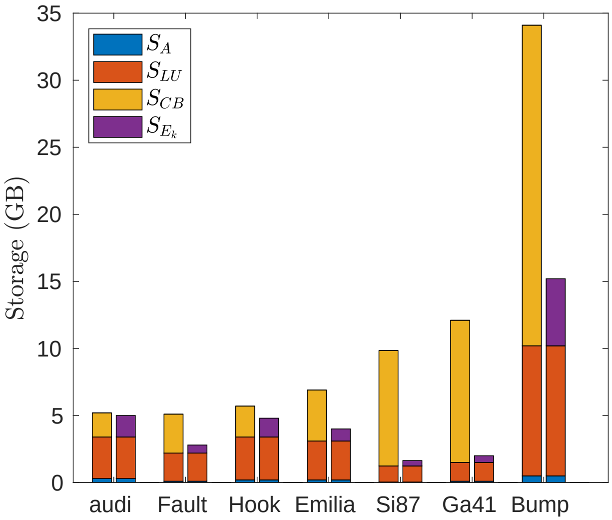
$\Rightarrow$ Total storage: $S_A + S_{LU} + \max(S_{CB}, S_{E_k})$

If $S_{E_k} \leq S_{CB}$, zero storage overhead!

Storage overhead: results



Storage overhead: results



⇒ zero storage overhead on all matrices

Some ingredients for the proof

The proof is based on *Stability of Block Algorithms with Fast Level-3 BLAS* (Demmel and Higham, 1992)

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

Inductive proof: easy to bound error of computing $S = A_{22} - L_{21}U_{12}$ and error of $S = L_{22}U_{22}$ is obtained by induction

Some ingredients for the proof

The proof is based on *Stability of Block Algorithms with Fast Level-3 BLAS* (Demmel and Higham, 1992)

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

Inductive proof: easy to bound error of computing $S = A_{22} - L_{21}U_{12}$ and error of $S = L_{22}U_{22}$ is obtained by induction

For BLR, several specific difficulties arise:

- Need to bound error of low-rank product kernel: $C = \widetilde{A}\widetilde{B} = X_A \left(Y_A^T X_B\right) Y_B^T$
- Choice of norm matters: to obtain best constants possible, we need a consistent, unitarily invariant norm
- Global bound is obtained from blockwise bounds
  $\Rightarrow$ we work with the Frobenius norm