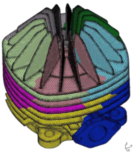
On the comparison of sparse multifrontal hierarchical and Block Low-Rank solvers

P. Amestoy[*,1] A. Buttari[*,2] P. Ghysels[‡] J.-Y. L'Excellent[†,3]
X. S. Li[‡] T. Mary[*,4] F.-H. Rouet[**]

[*]Université de Toulouse [‡]LBNL [†]ENS Lyon [**]LSTC
[1]INPT-IRIT [2]CNRS-IRIT [3]INRIA-LIP [4]UPS-IRIT

MUMPS User Days, Montbonnot Saint-Martin, Jun. 1-2, 2017

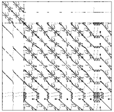
Introduction



Discretization of a physical problem
(e.g. Code_Aster, finite elements)

$\Downarrow$

A X = **B**, **A** large and sparse, **B** dense or sparse
Sparse direct methods : $\mathbf{A} = \mathbf{LU}$ $(\mathbf{LDL^T})$

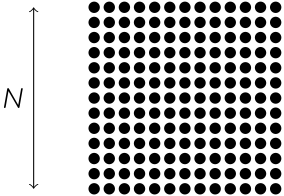


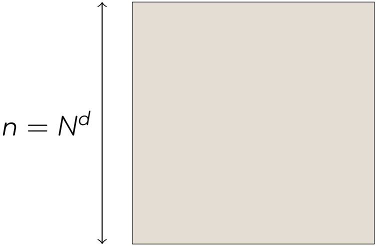
Often a significant part of simulation cost

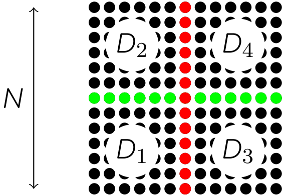
Objective discussed in this minisymposium:
how to reduce the cost of sparse direct solvers?

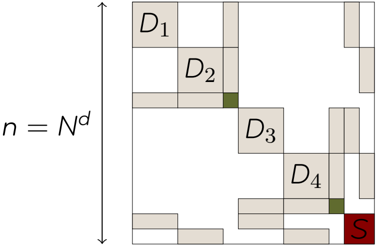
Focus on large-scale applications and architectures

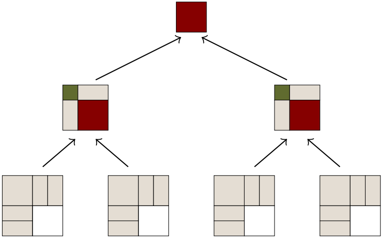
Multifrontal Factorization with Nested Dissection







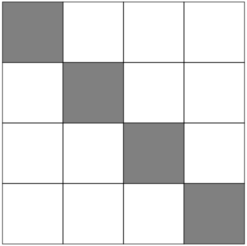




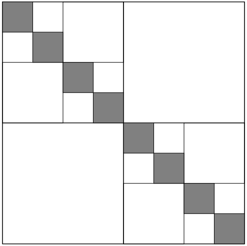
3D problem complexity

→ Flops: $O(n^2)$, mem: $O(n^{4/3})$

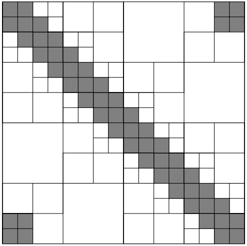
Low-rank matrix formats



BLR matrix

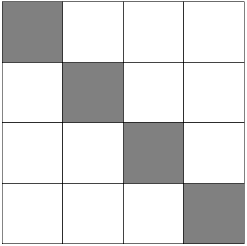


HODLR/HSS-matrix

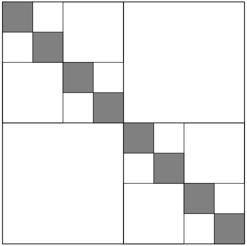


$\mathcal{H}/\mathcal{H}^2$-matrix

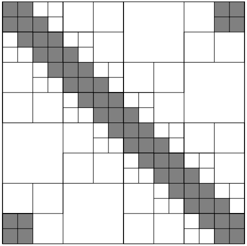
Low-rank matrix formats



BLR matrix



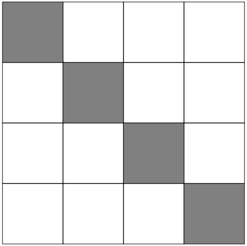
HODLR/HSS-matrix



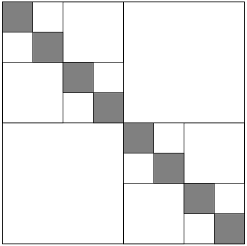
$\mathcal{H}/\mathcal{H}^2$-matrix

A block B represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

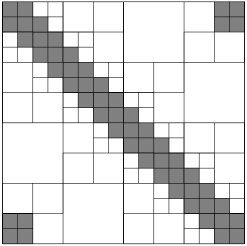
Low-rank matrix formats



BLR matrix



HODLR/HSS-matrix

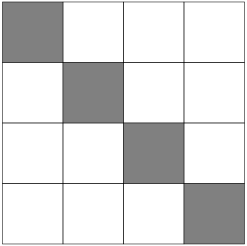


$\mathcal{H}/\mathcal{H}^2$-matrix

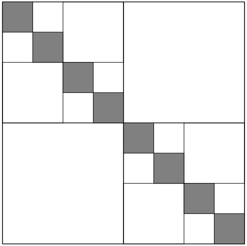
A block B represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.

Block-admissibility condition:

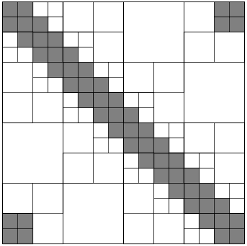
- Weak: $\sigma \times \tau$ is admissible $\Leftrightarrow \sigma \neq \tau$
- Strong: $\sigma \times \tau$ is admissible $\Leftrightarrow \mathrm{dist}(\sigma, \tau) > \eta \max(\mathrm{diam}(\sigma), \mathrm{diam}(\tau))$



BLR matrix



HODLR/HSS-matrix

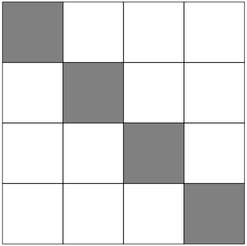


$\mathcal{H}/\mathcal{H}^2$-matrix

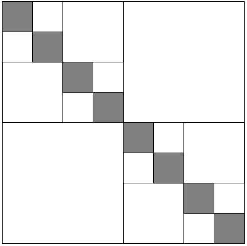
$$\tilde{B} = XY^T \text{ such that } \mathrm{rank}(\tilde{B}) = k_\varepsilon \text{ and } \|B - \tilde{B}\| \leq \varepsilon$$

If $k_\varepsilon \ll \mathrm{size}(B) \Rightarrow$ memory and flops can be reduced with a controlled loss of accuracy ($\leq \varepsilon$)

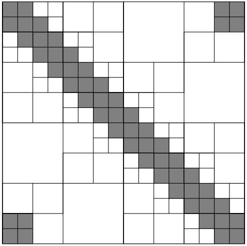
Low-rank matrix formats



BLR matrix

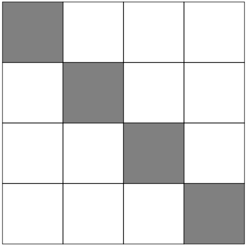


HODLR/HSS-matrix

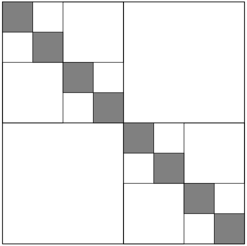


$\mathcal{H}/\mathcal{H}^2$-matrix

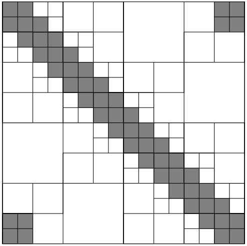
	BLR	HODLR	HSS	$\mathcal{H}$	$\mathcal{H}^2$
blocking	flat	hierar.	hierar.	hierar.	hierar.
adm. cond.	both	weak	weak	strong	strong
nested basis	no	no	yes	no	yes



BLR matrix



HODLR/HSS-matrix



$\mathcal{H}/\mathcal{H}^2$-matrix

Objective of this work: compare BLR and hierarchical formats, both from a theoretical and experimental standpoint

$\Rightarrow$ *collaboration between BLR-based* **MUMPS** *and HSS-based* **STRUMPACK** *teams.*

Main differences between MUMPS and STRUMPACK

Full-Rank Solvers

- Both are multifrontal

Full-Rank Solvers

- Both are multifrontal
- STRUMPACK supports LU only $\Rightarrow$ experiments are all performed on unsymmetric matrices

Full-Rank Solvers

- Both are multifrontal
- STRUMPACK supports LU only $\Rightarrow$ experiments are all performed on unsymmetric matrices
- STRUMPACK pivots inside diagonal blocks only; MUMPS has several options and was used with restricted pivoting too

Full-Rank Solvers

- Both are multifrontal
- STRUMPACK supports LU only $\Rightarrow$ experiments are all performed on unsymmetric matrices
- STRUMPACK pivots inside diagonal blocks only; MUMPS has several options and was used with restricted pivoting too
- Both support geometric and algebraic orderings: `METIS 5.1.0` is used in the experiments

Full-Rank Solvers

- Both are multifrontal
- STRUMPACK supports LU only $\Rightarrow$ experiments are all performed on unsymmetric matrices
- STRUMPACK pivots inside diagonal blocks only; MUMPS has several options and was used with restricted pivoting too
- Both support geometric and algebraic orderings: `METIS 5.1.0` is used in the experiments
- Both can exploit both shared- and distributed-memory architectures:
  - Shared-memory MUMPS: mainly node // based on multithreaded BLAS and OpenMP + some experimental tree // in OpenMP
  - Shared-memory STRUMPACK: tree and node // in handcoded OpenMP (sequential BLAS)
  - Distributed-memory MUMPS: tree MPI // + node 1D MPI //
  - Distributed-memory STRUMPACK: tree MPI // + node 2D MPI //

Low-Rank Solvers

- MUMPS uses BLR, STRUMPACK uses HSS

Low-Rank Solvers

- MUMPS uses BLR, STRUMPACK uses HSS
- Factorization algorithm:
  - MUMPS interleaves compressions and factorizations of panels
  - STRUMPACK first compresses the entire matrix, then performs a ULV factorization

⇒ STRUMPACK is fully-structured while MUMPS is not

Low-Rank Solvers

- MUMPS uses BLR, STRUMPACK uses HSS
- Factorization algorithm:
  - MUMPS interleaves compressions and factorizations of panels
  - STRUMPACK first compresses the entire matrix, then performs a ULV factorization

  ⇒ STRUMPACK is fully-structured while MUMPS is not
- Compression:
  - Kernel: both use truncated QR with column pivoting, with in addition random sampling in STRUMPACK
  - Threshold: absolute in MUMPS, relative in STRUMPACK

Low-Rank Solvers

- MUMPS uses BLR, STRUMPACK uses HSS
- Factorization algorithm:
  - MUMPS interleaves compressions and factorizations of panels
  - STRUMPACK first compresses the entire matrix, then performs a ULV factorization

  ⇒ STRUMPACK is fully-structured while MUMPS is not
- Compression:
  - Kernel: both use truncated QR with column pivoting, with in addition random sampling in STRUMPACK
  - Threshold: absolute in MUMPS, relative in STRUMPACK
- Assembly (extend-add):
  - contribution block not compressed in MUMPS ⇒ FR assembly
  - contribution block compressed in STRUMPACK ⇒ LR assembly

Low-Rank Solvers

- MUMPS uses BLR, STRUMPACK uses HSS
- Factorization algorithm:
  - MUMPS interleaves compressions and factorizations of panels
  - STRUMPACK first compresses the entire matrix, then performs a ULV factorization

  $\Rightarrow$ STRUMPACK is fully-structured while MUMPS is not
- Compression:
  - Kernel: both use truncated QR with column pivoting, with in addition random sampling in STRUMPACK
  - Threshold: absolute in MUMPS, relative in STRUMPACK
- Assembly (extend-add):
  - contribution block not compressed in MUMPS $\Rightarrow$ FR assembly
  - contribution block compressed in STRUMPACK $\Rightarrow$ LR assembly
- Both only compress fronts of size ≥ 1000

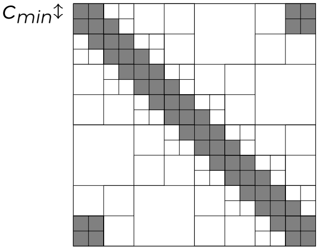
Low-Rank Solvers

- MUMPS uses BLR, STRUMPACK uses HSS
- Factorization algorithm:
  - MUMPS interleaves compressions and factorizations of panels
  - STRUMPACK first compresses the entire matrix, then performs a ULV factorization

  ⇒ STRUMPACK is fully-structured while MUMPS is not
- Compression:
  - Kernel: both use truncated QR with column pivoting, with in addition random sampling in STRUMPACK
  - Threshold: absolute in MUMPS, relative in STRUMPACK
- Assembly (extend-add):
  - contribution block not compressed in MUMPS ⇒ FR assembly
  - contribution block compressed in STRUMPACK ⇒ LR assembly
- Both only compress fronts of size ≥ 1000
- Solution phase:
  - BLR solve not yet available in MUMPS ⇒ performed in FR
  - HSS solve available in STRUMPACK

Complexity of the factorization

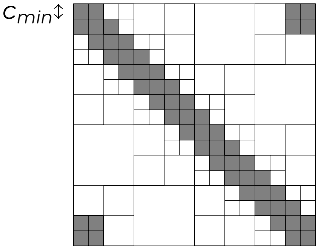
$\mathcal{H}$-admissibility and sparsity constant



- $\mathcal{H}$-admissibility condition: A partition $P \in \mathcal{P}(\mathcal{I} \times \mathcal{I})$ is admissible iff

$$\forall \sigma \times \tau \in P, \quad \sigma \times \tau \text{ is admissible} \quad \text{or} \quad \min(\#\sigma, \#\tau) \leq c_{min}$$

$\mathcal{H}$-admissibility and sparsity constant



(here, $c_{sp} = 6$)

- $\mathcal{H}$-admissibility condition: A partition $P \in \mathcal{P}(\mathcal{I} \times \mathcal{I})$ is admissible iff

$$\forall \sigma \times \tau \in P, \quad \sigma \times \tau \text{ is admissible} \quad \text{or} \quad \min(\#\sigma, \#\tau) \leq c_{min}$$

- The sparsity constant c_{sp} is defined as the maximal number of blocks of the same size on a given row or column. It measures the sparsity of the blocking imposed by the partition P.
  - In BLR, fully refined blocking $\Rightarrow$ c_{sp} = number of blocks per row
  - Can construct an admissible $\mathcal{H}$-partitioning such that $c_{sp} = O(1)$

$\mathcal{H}$ vs. BLR complexity

Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = O(mc_{sp}^2 r_{max}^2 \log^2 m)$ for $\mathcal{H}$ and $O(mc_{sp}^2 r_{max}^2)$ for HSS

m	matrix size
c_{sp}	sparsity constant
r_{max}	bound on the maximal rank of all blocks

$\mathcal{H}$ vs. BLR complexity

Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = O(mc_{sp}^2 r_{max}^2 \log^2 m)$ for $\mathcal{H}$ and $O(mc_{sp}^2 r_{max}^2)$ for HSS

m	matrix size
c_{sp}	sparsity constant
r_{max}	bound on the maximal rank of all blocks

	$\mathcal{H}$	HSS	BLR
c_{sp}			
r_{max}			
$\mathcal{C}_{facto}$			

$\mathcal{H}$ vs. BLR complexity

Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = O(mc_{sp}^2 r_{max}^2 \log^2 m)$ for $\mathcal{H}$ and $O(mc_{sp}^2 r_{max}^2)$ for HSS

m	matrix size
c_{sp}	sparsity constant
r_{max}	bound on the maximal rank of all blocks

	$\mathcal{H}$	HSS	BLR
c_{sp}	$O(1)^*$	$O(1)^*$	
r_{max}			
$\mathcal{C}_{facto}$			

Grasedyck & Hackbusch, 2003

$\mathcal{H}$ vs. BLR complexity

Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = O(mc_{sp}^2 r_{max}^2 \log^2 m)$ for $\mathcal{H}$ and $O(mc_{sp}^2 r_{max}^2)$ for HSS

m	matrix size
c_{sp}	sparsity constant
r_{max}	bound on the maximal rank of all blocks

	$\mathcal{H}$	HSS	BLR
c_{sp}	$O(1)^*$	$O(1)^*$	
r_{max}	small†	small‡	
$\mathcal{C}_{facto}$			

* *Grasedyck & Hackbusch, 2003*

† *Bebendorf & Hackbusch, 2003*

‡ *Chandrasekaran et al, 2010; Engquist & Ying, 2011*

$\mathcal{H}$ vs. BLR complexity

Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = O(mc_{sp}^2 r_{max}^2 \log^2 m)$ for $\mathcal{H}$ and $O(mc_{sp}^2 r_{max}^2)$ for HSS

m	matrix size
c_{sp}	sparsity constant
r_{max}	bound on the maximal rank of all blocks

	$\mathcal{H}$	HSS	BLR
c_{sp}	$O(1)^*$	$O(1)^*$	
r_{max}	small[†]	small[‡]	
$\mathcal{C}_{facto}$	$O(r_{max}^2 m \log^2 m)$	$O(r_{max}^2 m)$	

[*] *Grasedyck & Hackbusch, 2003*

[†] *Bebendorf & Hackbusch, 2003*

[‡] *Chandrasekaran et al, 2010; Engquist & Ying, 2011*

$\mathcal{H}$ vs. BLR complexity

Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = O(mc_{sp}^2 r_{max}^2 \log^2 m)$ for $\mathcal{H}$ and $O(mc_{sp}^2 r_{max}^2)$ for HSS

m	matrix size
c_{sp}	sparsity constant
r_{max}	bound on the maximal rank of all blocks

	$\mathcal{H}$	HSS	BLR
c_{sp}	$O(1)^*$	$O(1)^*$	m/b
r_{max}	small†	small‡	
$\mathcal{C}_{facto}$	$O(r_{max}^2 m \log^2 m)$	$O(r_{max}^2 m)$	

Grasedyck & Hackbusch, 2003

†*Bebendorf & Hackbusch, 2003*

‡*Chandrasekaran et al, 2010; Engquist & Ying, 2011*

$\mathcal{H}$ vs. BLR complexity

Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = O(mc_{sp}^2 r_{max}^2 \log^2 m)$ for $\mathcal{H}$ and $O(mc_{sp}^2 r_{max}^2)$ for HSS

m	matrix size
c_{sp}	sparsity constant
r_{max}	bound on the maximal rank of all blocks

	$\mathcal{H}$	HSS	BLR
c_{sp}	$O(1)^*$	$O(1)^*$	m/b
r_{max}	small†	small‡	b
$\mathcal{C}_{facto}$	$O(r_{max}^2 m \log^2 m)$	$O(r_{max}^2 m)$	

**Grasedyck & Hackbusch, 2003*

†*Bebendorf & Hackbusch, 2003*

‡*Chandrasekaran et al, 2010; Engquist & Ying, 2011*

$\mathcal{H}$ vs. BLR complexity

Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = O(mc_{sp}^2 r_{max}^2 \log^2 m)$ for $\mathcal{H}$ and $O(mc_{sp}^2 r_{max}^2)$ for HSS

m	matrix size
c_{sp}	sparsity constant
r_{max}	bound on the maximal rank of all blocks

	$\mathcal{H}$	HSS	BLR
c_{sp}	$O(1)^*$	$O(1)^*$	m/b
r_{max}	small†	small‡	b
$\mathcal{C}_{facto}$	$O(r_{max}^2 m \log^2 m)$	$O(r_{max}^2 m)$	$O(m^3)$

* *Grasedyck & Hackbusch, 2003*

† *Bebendorf & Hackbusch, 2003*

‡ *Chandrasekaran et al, 2010; Engquist & Ying, 2011*

$\mathcal{H}$ vs. BLR complexity

Dense factorization complexity

Complexity: $\mathcal{C}_{facto} = O(mc_{sp}^2 r_{max}^2 \log^2 m)$ for $\mathcal{H}$ and $O(mc_{sp}^2 r_{max}^2)$ for HSS

m	matrix size
c_{sp}	sparsity constant
r_{max}	bound on the maximal rank of all blocks

	$\mathcal{H}$	HSS	BLR
c_{sp}	$O(1)^*$	$O(1)^*$	m/b
r_{max}	small†	small‡	b
$\mathcal{C}_{facto}$	$O(r_{max}^2 m \log^2 m)$	$O(r_{max}^2 m)$	$O(m^3)$

* *Grasedyck & Hackbusch, 2003*

† *Bebendorf & Hackbusch, 2003*

‡ *Chandrasekaran et al, 2010; Engquist & Ying, 2011*

BLR: a particular case of $\mathcal{H}$?

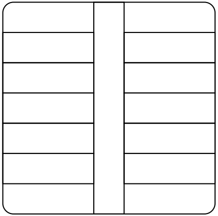
Problem: in $\mathcal{H}$ formalism, the maxrank of the blocks of a BLR matrix is $r_{max} = b$ (due to the non-admissible blocks)

Solution: bound the rank of the admissible blocks only, and make sure the non-admissible blocks are in small number

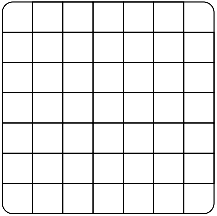
Complexity of dense BLR factorization

BLR-admissibility condition of a partition $\mathcal{P}$

$\mathcal{P}$ is admissible $\Leftrightarrow N_{na} = \#\{\sigma \times \tau \in \mathcal{P}, \;\; \sigma \times \tau \text{ is not admissible}\} \leq q$



Non-Admissible

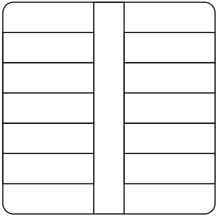


Admissible

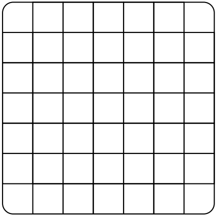
Complexity of dense BLR factorization

BLR-admissibility condition of a partition $\mathcal{P}$

$$\mathcal{P} \text{ is admissible } \Leftrightarrow N_{na} = \#\{\sigma \times \tau \in \mathcal{P},\ \ \sigma \times \tau \text{ is not admissible}\} \leq q$$



Non-Admissible



Admissible

Main result from *Amestoy et al, 2016*

There exists an admissible $\mathcal{P}$ for $q = O(1)$, s.t. the maxrank of the admissible blocks of A is $r = O(r^{\mathcal{H}}_{max})$

The dense factorization complexity thus becomes

$\mathcal{C}_{facto} = O(r^2m^3/b^2 + mb^2q^2) = O(r^2m^3/b^2 + mb^2) = O(rm^2)$ (for $b = O(\sqrt{rm})$)

Complexity of multifrontal BLR factorization

Under a nested dissection assumption, the sparse (multifrontal) complexity is directly obtained from the dense complexity

	operations (OPC)		factor size (NNZ)	
	$r = O(1)$	$r = O(N)$	$r = O(1)$	$r = O(N)$
FR	$O(n^2)$	$O(n^2)$	$O(n^{\frac{4}{3}})$	$O(n^{\frac{4}{3}})$
BLR	$O(n^{\frac{4}{3}})$	$O(n^{\frac{5}{3}})$	$O(n \log n)$	$O(n^{\frac{7}{6}} \log n)$
HSS	$O(n)$	$O(n^{\frac{4}{3}})$	$O(n)$	$O(n^{\frac{7}{6}})$

in the 3D case (similar analysis possible for 2D)

Experimental complexity: test problems

1. **`Poisson`**: N^3 grid with a 7-point stencil with $u = 1$ on the boundary $\partial\Omega$

$$\Delta u = f$$

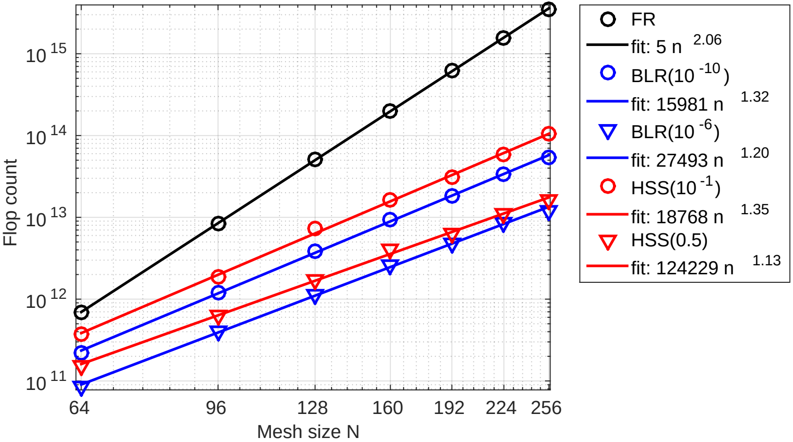
   Rank bound is $r_{max} = O(1)$ for BLR (and $\mathcal{H}$), and $r_{max} = O(N)$ for HSS.

2. **`Helmholtz`**: N^3 grid with a 27-point stencil, ω is the angular frequency, $v(x)$ is the seismic velocity field, and $u(x, \omega)$ is the time-harmonic wavefield solution to the forcing term $s(x, \omega)$.

$$\left(-\Delta - \frac{\omega^2}{v(x)^2}\right) u(x, \omega) = s(x, \omega)$$

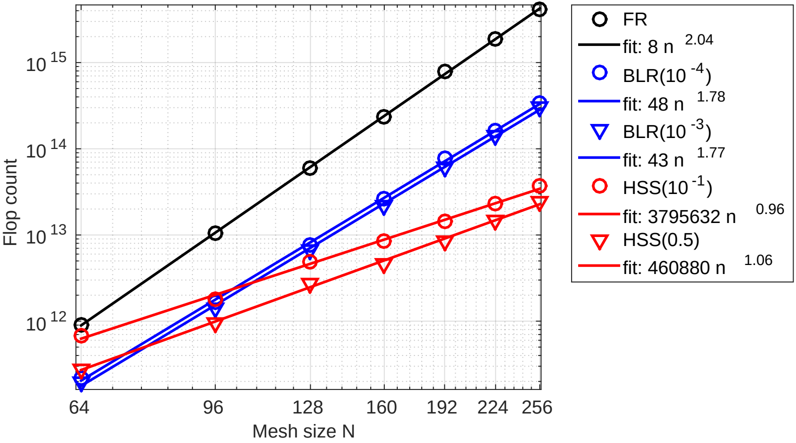
   ω is fixed and equal to 4Hz.
   Rank bound is $r_{max} = O(N)$ for both BLR and HSS.

Experimental flop complexity: Poisson



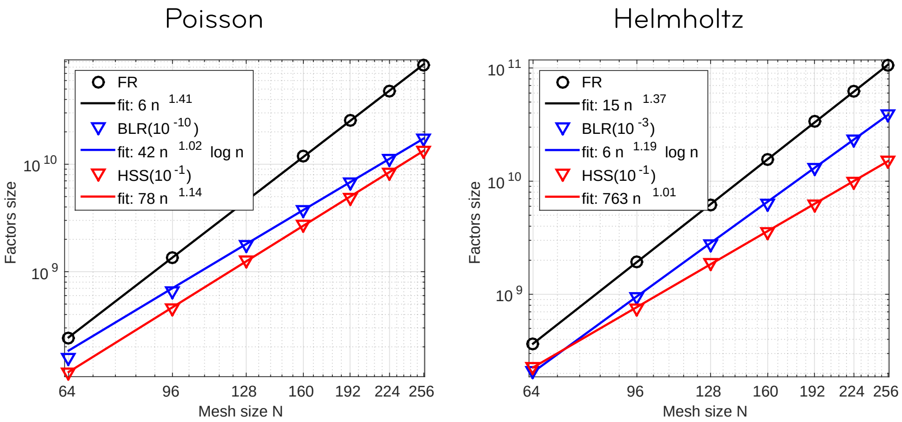
- good agreement with the theory ($O(n^{4/3})$ for both BLR and HSS)
- higher threshold leads to lower exponent:
  - relaxed rank pattern in HSS
  - zero-rank blocks in BLR

Experimental flop complexity: Helmholtz



- good agreement with the theory ($O(n^{5/3})$ for BLR, $O(n^{4/3})$ for HSS)
- threshold has almost no influence on the exponent

Experimental factor size complexity



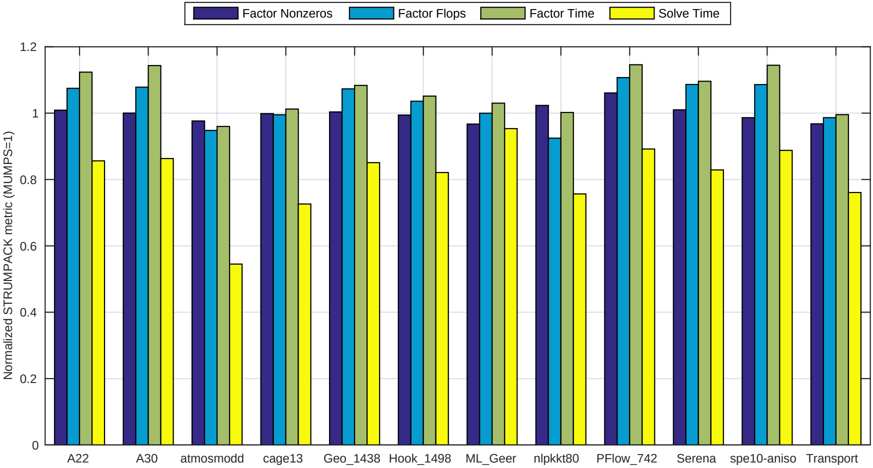
- good agreement with the theory
  - Poisson: $O(n \log n)$ for BLR, $O(n^{7/6})$ for HSS
  - Helmholtz: $O(n^{7/6} \log n)$ for BLR, $O(n^{7/6})$ for HSS

Preliminary performance results

Experimental Setting

- Experiments are done on the cori supercomputer of NERSC
  - Two Intel(r) 16-cores Haswell @ 2.3 GHz per node
  - Peak per core is 36.8 GF/s
  - Total memory per node is 128 GB
- Test problems come from several real-life applications: Seismic (5Hz), Electromagnetism (S3), Structural (perf008d, Geo_1438, Hook_1498, ML_Geer, Serena, Transport), CFD (atmosmodd, PFlow_742), MHD (A22, A30), Optimization (nlpkkt80), and Graph (cage13)
- We test 7 tolerance values (from 9e-1 to 1e-6) and FR, and compare the time for factorization + solve with:
  - 1 step of iterative refinement in FR
  - GMRES iterative solver in LR with required accuracy of 10^{-6} and restart of 30

Full-Rank solvers comparison



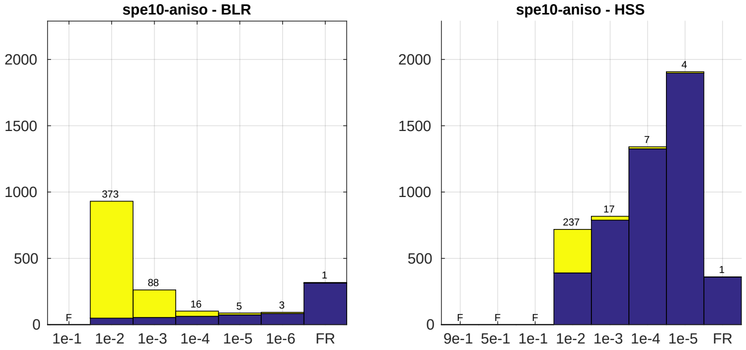
$\Rightarrow$ very similar FR performance

Preconditioner vs direct solver mode

Optimal tolerance choice

	BLR	HSS
A22	1e-5	FR
A30	1e-4	FR
atmosmodd	1e-4	9e-1
cage13	1e-1	9e-1
Geo_1438	1e-4	FR
Hook_1498	1e-5	FR
ML_Geer	1e-6	FR
nlpkkt80	1e-5	5e-1
PFlow_742	1e-6	FR
Serena	1e-4	1e-1
spe10-aniso	1e-5	FR
Transport	1e-5	FR

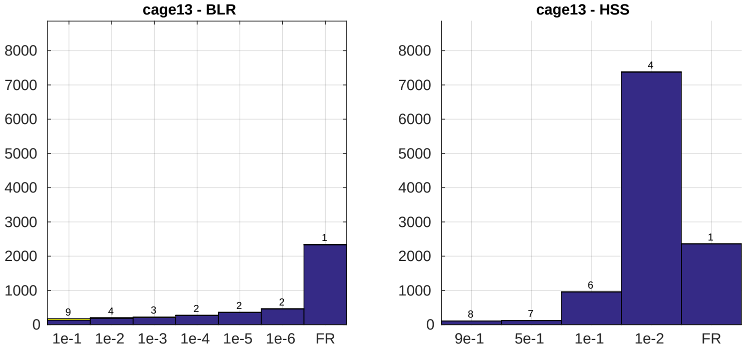
When high accuracy is needed...



spe10-aniso matrix

- No convergence except for low tolerances ⇒ direct solver mode is needed
- BLR is better suited as HSS rank is too high

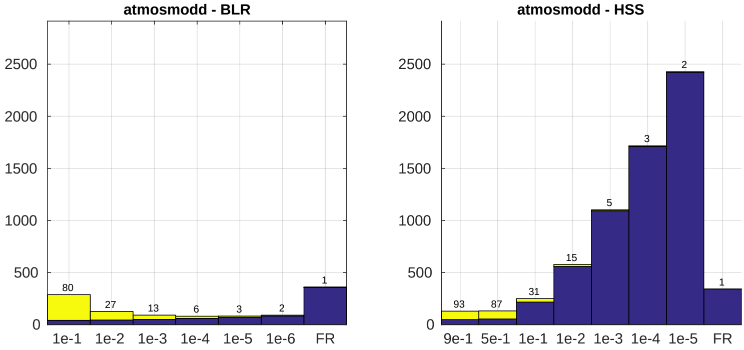
When preconditioning works well...



cage13 matrix

- Fast convergence even for high tolerance ⇒ preconditioner mode is better suited
- As the size grows, HSS will gain the upper hand

The middle ground



atmosmodd matrix

- Find compromise between accuracy and compression
- In general, BLR favors direct solver while HSS favors preconditioner mode

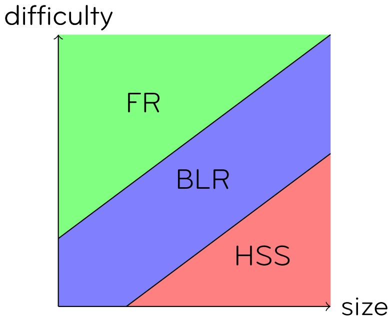
⇒ Performance comparison will depend on numerical difficulty and size of the problem

Preconditioner vs direct solver mode

Optimal tolerance choice

	BLR	HSS
A22	1e-5	FR
A30	1e-4	FR
atmosmodd	1e-4	9e-1
cage13	1e-1	9e-1
Geo_1438	1e-4	FR
Hook_1498	1e-5	FR
ML_Geer	1e-6	FR
nlpkkt80	1e-5	5e-1
PFlow_742	1e-6	FR
Serena	1e-4	1e-1
spe10-aniso	1e-5	FR
Transport	1e-5	FR

These preliminary results seem to suggest the following trend:

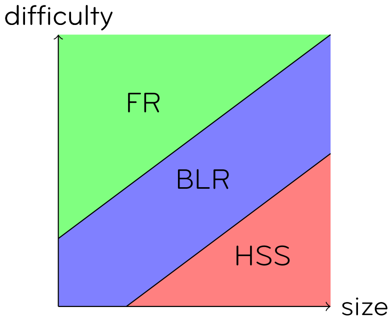


Preconditioner vs direct solver mode

Optimal tolerance choice

	BLR	HSS
A22	1e-5	FR
A30	1e-4	FR
atmosmodd	1e-4	9e-1
cage13	1e-1	9e-1
Geo_1438	1e-4	FR
Hook_1498	1e-5	FR
ML_Geer	1e-6	FR
nlpkkt80	1e-5	5e-1
PFlow_742	1e-6	FR
Serena	1e-4	1e-1
spe10-aniso	1e-5	FR
Transport	1e-5	FR

These preliminary results seem to suggest the following trend:



⇒ much further work needed to confirm this trend and to fully understand the differences between low-rank formats

References and acknowledgements

Software packages

- MUMPS 5.1.0 (including BLR factorization for the first time)
- STRUMPACK-dense-1.1.1 and STRUMPACK-sparse 1.1.0

References

- Amestoy, Ashcraft, Boiteau, Buttari, L'Excellent, and Weisbecker. *Improving Multifrontal Methods by means of Block Low-Rank Representations*, SIAM SISC, 2015.
- Amestoy, Buttari, L'Excellent, and Mary. *On the Complexity of the Block Low-Rank Multifrontal Factorization*, SIAM SISC, 2017.
- Amestoy, Buttari, L'Excellent, and Mary. *Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures*, submitted to TOMS.
- Ghysels, Li, Rouet, Williams, Napov. *An efficient multi-core implementation of a novel HSS-structured multifrontal solver using randomized sampling*, SIAM SISC, 2015.
- Rouet, Li, Ghysels, Napov. *A distributed-memory package for dense hierarchically semi-separable matrix computations using randomization*, ACM TOMS, 2016.

Acknowledgements

- NERSC for providing access to the machine
- EMGS, SEISCOPE, EDF, and LBNL for providing the matrices

Thanks!

Questions?