Comparison of BLR and HSS Low-Rank Formats in Multifrontal Solvers: Theory and Practice

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
1INPT-IRIT  2CNRS-IRIT  3INRIA-LIP  4UPS-IRIT

CSE’17, Atlanta, Feb. 27 - Mar. 3
Introduction
Sparse direct solvers

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

\[ \mathbf{A} \mathbf{X} = \mathbf{B}, \ \mathbf{A} \text{ large and sparse, } \mathbf{B} \text{ dense or sparse} \]

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

\[ n = N^d \]
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
A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$. If they have a small diameter and are far away their interaction is weak $\Rightarrow$ rank is low.
A block $B$ represents the interaction between two subdomains $\sigma$ and $\tau$. 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 $\iff \sigma \neq \tau$
- **Strong:** $\sigma \times \tau$ is admissible $\iff \text{dist}(\sigma, \tau) > \eta \max(\text{diam}(\sigma), \text{diam}(\tau))$
Low-rank matrix formats

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

If \( k_\varepsilon \ll \text{size}(B) \Rightarrow \text{memory and flops can be reduced with a controlled loss of accuracy (} \leq \varepsilon \)\)
Low-rank matrix formats

<table>
<thead>
<tr>
<th></th>
<th>BLR</th>
<th>HODLR</th>
<th>HSS</th>
<th>$\mathcal{H}$</th>
<th>$\mathcal{H}^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>blocking</td>
<td>flat</td>
<td>hierar.</td>
<td>hierar.</td>
<td>hierar.</td>
<td>hierar.</td>
</tr>
<tr>
<td>adm. cond.</td>
<td>both</td>
<td>weak</td>
<td>weak</td>
<td>strong</td>
<td>strong</td>
</tr>
<tr>
<td>nested basis</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
Low-rank matrix formats

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

⇒ collaboration between BLR-based MUMPS and HSS-based STRUMPACK teams.
Main differences between MUMPS and STRUMPACK
• Both are multifrontal
• Both are multifrontal

• STRUMPACK supports $LU$ only $\Rightarrow$ experiments are all performed on unsymmetric matrices
• 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 handcodded 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
• 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
• 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 $\geq 1000$
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 \( \geq 1000 \)
- Solution phase:
  - BLR solve not yet available in MUMPS ⇒ performed in FR
  - HSS solve available in STRUMPACK
Complexity of the factorization
- **$\mathcal{H}$-admissibility condition**: A partition $P \in \mathcal{P}(I \times I)$ is admissible iff

\[ \forall \sigma \times \tau \in P, \ \sigma \times \tau \text{ is admissible} \quad \text{or} \quad \min(\#\sigma, \#\tau) \leq c_{min} \]
\( H \)-admissibility and sparsity constant

- **\( H \)-admissibility condition**: A partition \( P \in \mathcal{P}(I \times I) \) is admissible iff
  \[
  \forall \sigma \times \tau \in P, \quad \sigma \times \tau \text{ is admissible or } \min(\#\sigma, \#\tau) \leq c_{\text{min}}
  \]

- 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 \( \Rightarrow c_{sp} = \text{number of blocks per row} \)
  - Can construct an admissible \( H \)-partitioning such that \( c_{sp} = O(1) \)
Dense factorization complexity

| Complexity: $C_{\text{facto}} = O(m c_{sp}^2 r_{max}^2 \log^2 m)$ for $\mathcal{H}$ and $O(m c_{sp}^2 r_{max}^2)$ for HSS |
|---|---|
| $m$ | matrix size |
| $c_{sp}$ | sparsity constant |
| $r_{max}$ | bound on the maximal rank of all blocks |
## Dense factorization complexity

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

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

<table>
<thead>
<tr>
<th></th>
<th>$\mathcal{H}$</th>
<th>HSS</th>
<th>BLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{sp}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{max}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{\text{facto}}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Dense factorization complexity

**Complexity:** \( C_{\text{facto}} = O(mc_{sp}^2 r_{max}^2 \log^2 m) \) 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

<table>
<thead>
<tr>
<th></th>
<th>( \mathcal{H} )</th>
<th>HSS</th>
<th>BLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_{sp} )</td>
<td>( O(1)^* )</td>
<td>( O(1)^* )</td>
<td>( O(1)^* )</td>
</tr>
<tr>
<td>( r_{max} )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( C_{\text{facto}} )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( ^* \) Grasedyck & Hackbusch, 2003
### Dense factorization complexity

**Complexity:** \( C_{\text{facto}} = O(m c_{sp}^2 r_{max}^2 \log^2 m) \) for \( \mathcal{H} \) and \( O(m c_{sp}^2 r_{max}^2) \) for HSS

- **\( m \)**: matrix size
- **\( c_{sp} \)**: sparsity constant
- **\( r_{max} \)**: bound on the maximal rank of all blocks

<table>
<thead>
<tr>
<th></th>
<th>( \mathcal{H} )</th>
<th>HSS</th>
<th>BLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_{sp} )</td>
<td>( O(1)^* )</td>
<td>( O(1)^* )</td>
<td></td>
</tr>
<tr>
<td>( r_{max} )</td>
<td>small†</td>
<td>small‡</td>
<td></td>
</tr>
<tr>
<td>( C_{\text{facto}} )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Grasedyck & Hackbusch, 2003*

†Bebendorf & Hackbusch, 2003

‡Chandrasekaran et al, 2010; Engquist & Ying, 2011
### Dense factorization complexity

**Complexity:** \( C_{\text{facto}} = O(mc_{sp}^2 r_{max}^2 \log^2 m) \) for \( \mathcal{H} \) and \( O(mc_{sp}^2 r_{max}) \) for HSS

- \( m \): matrix size
- \( c_{sp} \): sparsity constant
- \( r_{max} \): bound on the maximal rank of all blocks

<table>
<thead>
<tr>
<th></th>
<th>( \mathcal{H} )</th>
<th>HSS</th>
<th>BLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_{sp} )</td>
<td>( O(1)^* )</td>
<td>( O(1)^* )</td>
<td></td>
</tr>
<tr>
<td>( r_{max} )</td>
<td>small(^\dagger)</td>
<td>small(^\ddagger)</td>
<td></td>
</tr>
<tr>
<td>( C_{\text{facto}} )</td>
<td>( O(r_{max}^2 m \log^2 m) )</td>
<td>( O(r_{max}^2 m) )</td>
<td></td>
</tr>
</tbody>
</table>

*Grasedyck & Hackbusch, 2003*

\(^\dagger\)Bebendorf & Hackbusch, 2003

\(^\ddagger\)Chandrasekaran et al, 2010; Engquist & Ying, 2011
**H vs. BLR complexity**

### Dense factorization complexity

**Complexity:** \( C_{\text{facto}} = O(m c_{sp}^2 r_{max}^2 \log^2 m) \) for \( H \) and \( O(m c_{sp}^2 r_{max}^2) \) for HSS

- **\( m \)** matrix size
- **\( c_{sp} \)** sparsity constant
- **\( r_{max} \)** bound on the maximal rank of all blocks

<table>
<thead>
<tr>
<th></th>
<th>( H )</th>
<th>HSS</th>
<th>BLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_{sp} )</td>
<td>( O(1)^* )</td>
<td>( O(1)^* )</td>
<td>( m/b )</td>
</tr>
<tr>
<td>( r_{max} )</td>
<td>small(^\dagger)</td>
<td>small(^\dagger)</td>
<td></td>
</tr>
<tr>
<td>( C_{\text{facto}} )</td>
<td>( O(r_{max}^2 m \log^2 m) )</td>
<td>( O(r_{max}^2 m) )</td>
<td></td>
</tr>
</tbody>
</table>

\(^*\) Grasedyck & Hackbusch, 2003
\(^\dagger\) Bebendorf & Hackbusch, 2003
\(^\ddagger\) Chandrasekaran et al, 2010; Engquist & Ying, 2011
$\mathcal{H}$ vs. BLR complexity

### Dense factorization complexity

**Complexity:** $C_{\text{facto}} = O(m c_{sp}^2 r_{max}^2 \log^2 m)$ for $\mathcal{H}$ and $O(m c_{sp}^2 r_{max}^2)$ for HSS

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

### Comparison Table

<table>
<thead>
<tr>
<th></th>
<th>$\mathcal{H}$</th>
<th>HSS</th>
<th>BLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{sp}$</td>
<td>$O(1)^*$</td>
<td>$O(1)^*$</td>
<td>$m/b$</td>
</tr>
<tr>
<td>$r_{max}$</td>
<td>small$^\dagger$</td>
<td>small$^\ddagger$</td>
<td>$b$</td>
</tr>
<tr>
<td>$C_{\text{facto}}$</td>
<td>$O(r_{max}^2 m \log^2 m)$</td>
<td>$O(r_{max}^2 m)$</td>
<td></td>
</tr>
</tbody>
</table>

*Grasedyck & Hackbusch, 2003
$^\dagger$Bebendorf & Hackbusch, 2003
$^\ddagger$Chandrasekaran et al, 2010; Engquist & Ying, 2011
### Dense factorization complexity

**Complexity:** $C_{\text{facto}} = O(mc_{sp}^2 r_{max}^2 \log^2 m)$ 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

<table>
<thead>
<tr>
<th></th>
<th>$\mathcal{H}$</th>
<th>HSS</th>
<th>BLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{sp}$</td>
<td>$O(1)^*$</td>
<td>$O(1)^*$</td>
<td>$m/b$</td>
</tr>
<tr>
<td>$r_{max}$</td>
<td>small$\dagger$</td>
<td>small$\ddagger$</td>
<td>$b$</td>
</tr>
<tr>
<td>$C_{\text{facto}}$</td>
<td>$O(r_{max}^2 m \log^2 m)$</td>
<td>$O(r_{max}^2 m)$</td>
<td>$O(m^3)$</td>
</tr>
</tbody>
</table>

*Grasedyck & Hackbusch, 2003
†Bebendorf & Hackbusch, 2003
‡Chandrasekaran et al, 2010; Engquist & Ying, 2011
### Dense factorization complexity

**Complexity:** \( C_{\text{facto}} = O(mc_{sp}^2 r_{\text{max}}^2 \log^2 m) \) for \( \mathcal{H} \) and \( O(mc_{sp}^2 r_{\text{max}}^2) \) for HSS

- \( m \): matrix size
- \( c_{sp} \): sparsity constant
- \( r_{\text{max}} \): bound on the maximal rank of all blocks

<table>
<thead>
<tr>
<th></th>
<th>( \mathcal{H} )</th>
<th>HSS</th>
<th>BLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_{sp} )</td>
<td>( O(1)^* )</td>
<td>( O(1)^* )</td>
<td>( m/b )</td>
</tr>
<tr>
<td>( r_{\text{max}} )</td>
<td>small(^\dagger)</td>
<td>small(^\ddagger)</td>
<td>( b )</td>
</tr>
<tr>
<td>( C_{\text{facto}} )</td>
<td>( O(r_{\text{max}}^2 m \log^2 m) )</td>
<td>( O(r_{\text{max}}^2 m) )</td>
<td>( O(m^3) )</td>
</tr>
</tbody>
</table>

\(^*\text{Grasedyck \& Hackbusch, 2003}\)

\(^\dagger\text{Bebendorf \& Hackbusch, 2003}\)

\(^\ddagger\text{Chandrasekaran et al, 2010; Engquist \& Ying, 2011}\)

### BLR: a particular case of \( \mathcal{H} \)?

**Problem:** in \( \mathcal{H} \) formalism, the maxrank of the blocks of a BLR matrix is \( r_{\text{max}} = b \) (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
BLR-admissibility condition of a partition $\mathcal{P}$

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

Non-Admissible

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

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

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 $r = O(r_{\text{max}}^H)$

The dense factorization complexity thus becomes

$C_{\text{facto}} = O(r^2 m^3 / b^2 + m b^2 q^2) = O(r^2 m^3 / b^2 + m b^2) = O(r m^2)$ (for $b = O(\sqrt{r m})$)
Under a nested dissection assumption, the sparse (multifrontal) complexity is directly obtained from the dense complexity.

<table>
<thead>
<tr>
<th></th>
<th>operations (OPC)</th>
<th>factor size (NNZ)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( r = O(1) )</td>
<td>( r = O(1) )</td>
</tr>
<tr>
<td>( r = O(N) )</td>
<td>( O(n^2) )</td>
<td>( O(n^{\frac{4}{3}}) )</td>
</tr>
<tr>
<td>( r = O(N) )</td>
<td>( O(n^2) )</td>
<td>( O(n^{\frac{4}{3}}) )</td>
</tr>
</tbody>
</table>

in the 3D case (similar analysis possible for 2D)
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, $\omega$ 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 $s(x, \omega)$.

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

$\omega$ is fixed and equal to 4Hz.

Rank bound is $r_{max} = O(N)$ for both BLR and HSS.
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
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, Serena, Transport), CFD (atmosmodd), MHD (A16, A22, A30), Optimization (nlpkkt80), and Graph (cage13)
  (Only partial results shown in next slides)

- 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
## Optimal tolerance choice

<table>
<thead>
<tr>
<th></th>
<th>BLR</th>
<th>HSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>atmosmodd</td>
<td>1e-4</td>
<td>9e-1</td>
</tr>
<tr>
<td>cage13</td>
<td>9e-1</td>
<td>9e-1</td>
</tr>
<tr>
<td>Geo_1438</td>
<td>1e-4</td>
<td>FR</td>
</tr>
<tr>
<td>ML_Geer</td>
<td>1e-6</td>
<td>1e-4</td>
</tr>
<tr>
<td>nlpkkt80</td>
<td>1e-5</td>
<td>9e-1</td>
</tr>
<tr>
<td>Serena</td>
<td>1e-4</td>
<td>9e-1</td>
</tr>
<tr>
<td>spe10-aniso</td>
<td>1e-5</td>
<td>FR</td>
</tr>
<tr>
<td>Transport</td>
<td>1e-5</td>
<td>FR</td>
</tr>
</tbody>
</table>
When preconditioning works well...

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

**cage13 matrix**
When high accuracy is needed...

spe10-aniso matrix

- No convergence except for low tolerances $\implies$ direct solver mode is needed
- BLR is better suited as HSS rank is too high
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
<table>
<thead>
<tr>
<th>Dataset</th>
<th>BLR</th>
<th>HSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>atmosmodd</td>
<td>1e-4</td>
<td>9e-1</td>
</tr>
<tr>
<td>cage13</td>
<td>9e-1</td>
<td>9e-1</td>
</tr>
<tr>
<td>Geo_1438</td>
<td>1e-4</td>
<td>FR</td>
</tr>
<tr>
<td>ML_Geer</td>
<td>1e-6</td>
<td>1e-4</td>
</tr>
<tr>
<td>nlpkkt80</td>
<td>1e-5</td>
<td>9e-1</td>
</tr>
<tr>
<td>Serena</td>
<td>1e-4</td>
<td>9e-1</td>
</tr>
<tr>
<td>spe10-aniso</td>
<td>1e-5</td>
<td>FR</td>
</tr>
<tr>
<td>Transport</td>
<td>1e-5</td>
<td>FR</td>
</tr>
</tbody>
</table>

These preliminary results seem to suggest the following trend:

- **FR** (High Difficulty) corresponds to larger problem sizes and higher tolerances in both BLR and HSS methods.
- **BLR** and **HSS** methods show different optimal tolerances for different datasets, indicating that the optimal tolerance choice depends on the problem size and difficulty.

Further work is needed to confirm this trend and to fully understand the differences between low-rank formats.
These preliminary results seem to suggest the following trend:

<table>
<thead>
<tr>
<th></th>
<th>BLR</th>
<th>HSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>atmosmodd</td>
<td>1e-4</td>
<td>9e-1</td>
</tr>
<tr>
<td>cage13</td>
<td>9e-1</td>
<td>9e-1</td>
</tr>
<tr>
<td>Geo_1438</td>
<td>1e-4</td>
<td>FR</td>
</tr>
<tr>
<td>ML_Geer</td>
<td>1e-6</td>
<td>1e-4</td>
</tr>
<tr>
<td>nlpkkt80</td>
<td>1e-5</td>
<td>9e-1</td>
</tr>
<tr>
<td>Serena</td>
<td>1e-4</td>
<td>9e-1</td>
</tr>
<tr>
<td>spe10-aniso</td>
<td>1e-5</td>
<td>FR</td>
</tr>
<tr>
<td>Transport</td>
<td>1e-5</td>
<td>FR</td>
</tr>
</tbody>
</table>

⇒ 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


Acknowledgements

- **NERSC** for providing access to the machine
- **EMGS, SEISCOPE, EDF, and LBNL** for providing the matrices
Thanks!
Questions?