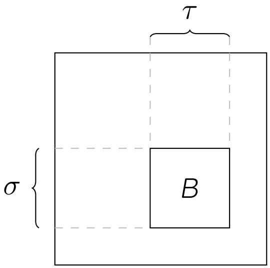
Exploiting Fast Matrix Arithmetic in Block Low-Rank Factorizations

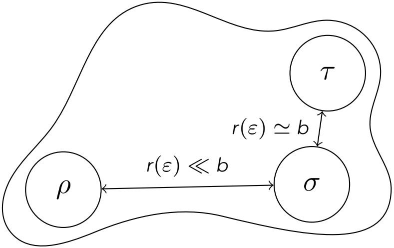
Theo Mary,
joint work with C.-P. Jeannerod, C. Pernet, D. Roche
University of Manchester, School of Mathematics

Structured Matrix Days 2019, Limoges

Data sparse matrices

In many $Ax = b$ applications, matrix A has a block low-rank structure





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

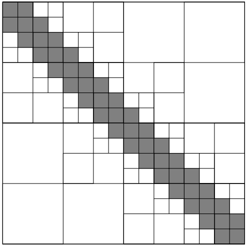
$$\underset{b \times b}{B} \approx \underset{b \times r(\varepsilon)}{X} \quad \underset{r(\varepsilon) \times b}{Y^T}$$

with $r(\varepsilon) \ll b$ such that $\|B - XY^T\| \leq \varepsilon$

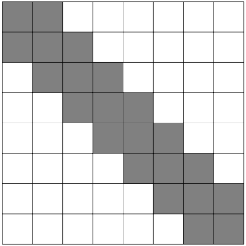
Flat vs hierarchical matrices

How to choose a good block partitioning of the matrix?

White: low-rank (LR) blocks (rank at most r)
Gray: full-rank (FR) blocks (stored exactly)



$\mathcal{H}$-matrix



BLR matrix

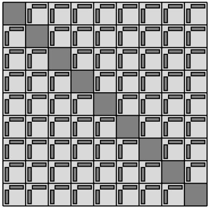
- $A = LU$ complexity $O(nr^2)$
- Hierarchical structure not well suited for parallel computing

- Simple, flat structure ideal for parallel computing
- Superlinear complexity $O(n^2 r)$

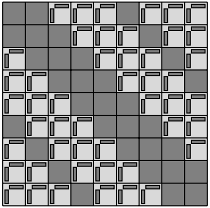
Objective of this work

- How to reduce the BLR complexity $O(n^2r)$ without losing its non hierarchical nature? Our idea: **use fast matrix arithmetic**?
- Fast algorithms can multiply $b \times b$ matrices in only $O(b^\omega)$ flops, with $2 < \omega < 3$, e.g. Strassen's algorithm $\Rightarrow \omega = \log_2 7 \approx 2.8$
- Simplifying assumption made for this talk: all off-diagonal blocks are LR $\Rightarrow$ generalization to $O(1)$ FR blocks per row/column in the paper

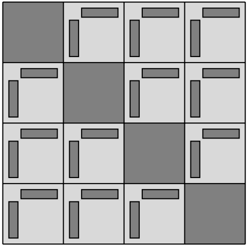
In this talk



In the paper

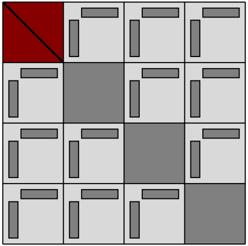


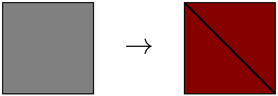
BLR matrix LU factorization: classical algorithm



Kernel costs:

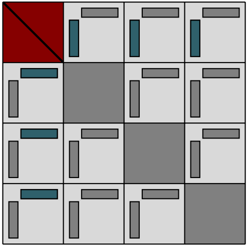
BLR matrix LU factorization: classical algorithm

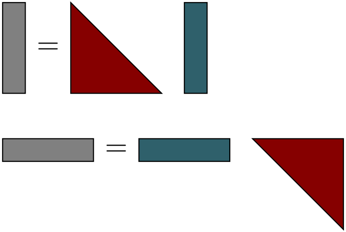




Kernel costs:

- **Factor** kernel: $O(b^3)$

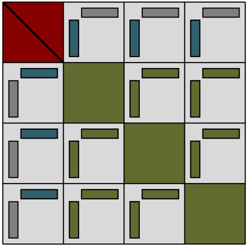


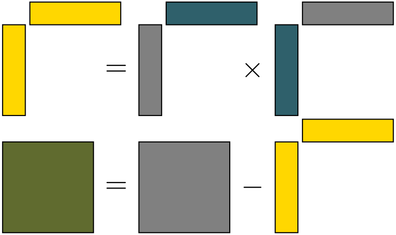


Kernel costs:

- **Factor** kernel: $O(b^3)$
- **Solve** kernel: $O(b^2 r)$

BLR matrix LU factorization: classical algorithm

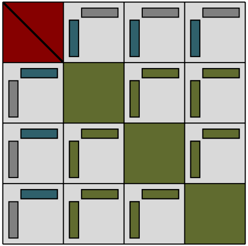


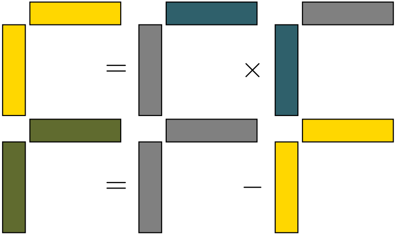


Kernel costs:

- **Factor** kernel: $O(b^3)$
- **Solve** kernel: $O(b^2 r)$
- **Update** kernel:
  - FR target: $O(b^2 r)$

BLR matrix LU factorization: classical algorithm

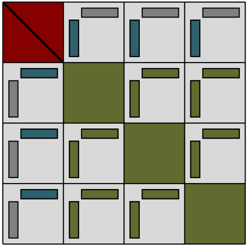




Kernel costs:

- **Factor** kernel: $O(b^3)$
- **Solve** kernel: $O(b^2 r)$
- **Update** kernel:
  - FR target: $O(b^2 r)$
  - LR target: $O(br^2)$

BLR matrix LU factorization: classical algorithm



Kernel costs:

- **Factor** kernel: $O(b^3) \to O(b^\omega)$
- **Solve** kernel: $O(b^2 r) \to O(b^2 r^{\omega-2})$
- **Update** kernel:
  - FR target: $O(b^2 r) \to O(b^2 r^{\omega-2})$
  - LR target: $O(br^2) \to O(br^{\omega-1})$

Complexity of the classical algorithm

- Let $p = n/b$ be the number of blocks per row/column
- Then the classical factorization algorithm costs:
  - $O(p)$ **Factor** kernel calls
  - $O(p^2)$ **Solve** kernel calls
  - $O(p^2)$ FR-**Update** and $O(p^3)$ LR-**Update** kernel calls

Complexity of the classical algorithm

- Let $p = n/b$ be the number of blocks per row/column
- Then the classical factorization algorithm costs:
  - $O(p)$ **Factor** kernel calls
  - $O(p^2)$ **Solve** kernel calls
  - $O(p^2)$ FR-**Update** and $O(p^3)$ LR-**Update** kernel calls
- Total:

$$O(pb^\omega + p^2b^2r^{\omega-2} + p^3br^{\omega-1})$$

Complexity of the classical algorithm

- Let $p = n/b$ be the number of blocks per row/column
- Then the classical factorization algorithm costs:
  - $O(p)$ **Factor** kernel calls
  - $O(p^2)$ **Solve** kernel calls
  - $O(p^2)$ FR-**Update** and $O(p^3)$ LR-**Update** kernel calls
- Total:

$$O(pb^\omega + p^2b^2r^{\omega-2} + p^3br^{\omega-1})$$

$$\subset O(n^2r^{\omega-2}) \qquad \text{for } b = \Theta((nr)^{1/2})$$

Complexity of the classical algorithm

- Let $p = n/b$ be the number of blocks per row/column
- Then the classical factorization algorithm costs:
  - $O(p)$ **Factor** kernel calls
  - $O(p^2)$ **Solve** kernel calls
  - $O(p^2)$ FR-**Update** and $O(p^3)$ LR-**Update** kernel calls
- Total:

$$O(pb^\omega + p^2b^2r^{\omega-2} + p^3br^{\omega-1})$$
$$\subset O(n^2r^{\omega-2}) \qquad \text{for } b = \Theta((nr)^{1/2})$$

- Reduction by a factor $O(r^{3-\omega}) \Rightarrow$ **underwhelming since** $n \gg r$

Complexity of the classical algorithm

- Let $p = n/b$ be the number of blocks per row/column
- Then the classical factorization algorithm costs:
  - $O(p)$ **Factor** kernel calls
  - $O(p^2)$ **Solve** kernel calls
  - $O(p^2)$ FR-**Update** and $O(p^3)$ LR-**Update** kernel calls
- Total:

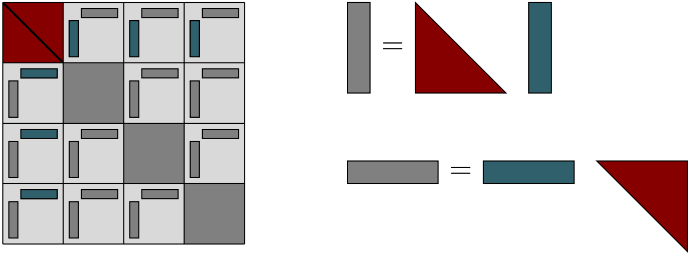
$$O(pb^\omega + p^2 b^2 r^{\omega-2} + p^3 b r^{\omega-1})$$

$$\subset O(n^2 r^{\omega-2}) \qquad \text{for } b = \Theta((nr)^{1/2})$$

- Reduction by a factor $O(r^{3-\omega})$ $\Rightarrow$ **underwhelming since** $n \gg r$
- The issue lies with the low granularity of LR computations
  - Factor kernel: $O(b^3) \to O(b^\omega)$
  - Solve kernel: $O(b^2 r) \to O(b^2 r^{\omega-2})$
  - FR-Update kernel: $O(b^2 r) \to O(b^2 r^{\omega-2})$
  - LR-Update kernel: $O(br^2) \to O(br^{\omega-1})$

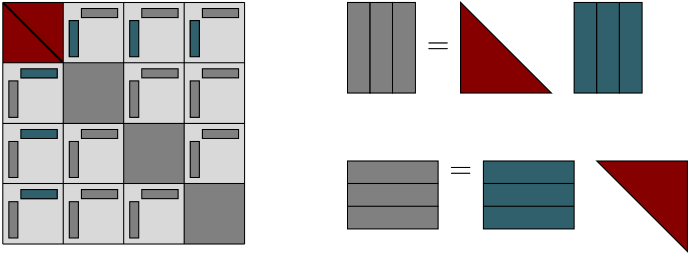
$\Rightarrow$ Can we obtain a complexity **subquadratic** in n?

New Solve kernel



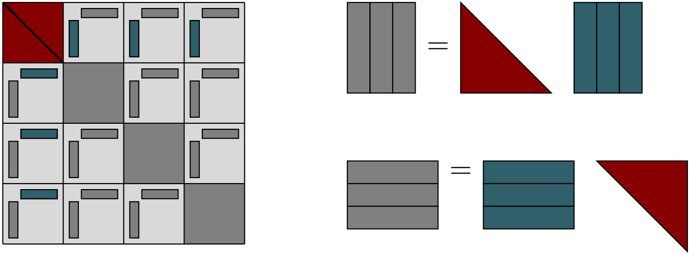
- Classical: $O(p^2)$ calls of cost $O(b^2 r^{\omega-2}) \Rightarrow O(p^2 b^2 r^{\omega-2})$

New Solve kernel



- Classical: $O(p^2)$ calls of cost $O(b^2 r^{\omega-2}) \Rightarrow O(p^2 b^2 r^{\omega-2})$
- New: $O(p)$ calls of cost $O(b^{\omega-1} pr) \Rightarrow O(p^2 b^{\omega-1} r)$

New Solve kernel

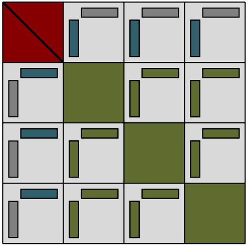


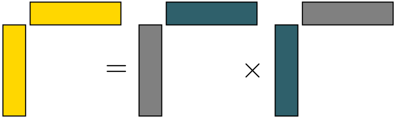
- Classical: $O(p^2)$ calls of cost $O(b^2 r^{\omega-2}) \Rightarrow O(p^2 b^2 r^{\omega-2})$
- New: $O(p)$ calls of cost $O(b^{\omega-1} pr) \Rightarrow O(p^2 b^{\omega-1} r)$

$\Rightarrow$ Reduction by a factor $O((b/r)^{3-\omega})$:

- no asymptotic gain if $\omega = 3$ (we only rearranged computations)
- no gain if $r \sim b$ (good enough granularity... but $O(n^\omega)$ complexity)
- possibly large asymptotic gain in general!

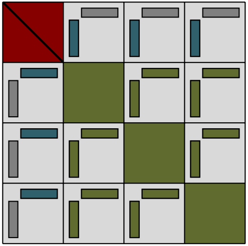
New Update kernel





- Classical product: $O(p^3)$ calls of cost $O(br^{\omega-1}) \Rightarrow O(p^3br^{\omega-1})$

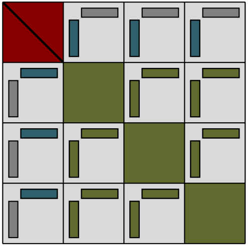
New Update kernel

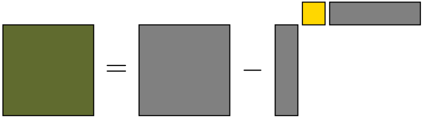


- Classical product: $O(p^3)$ calls of cost $O(br^{\omega-1}) \Rightarrow O(p^3 br^{\omega-1})$
- New product: $O(p)$ calls of cost $O(p^2 b^{\omega-2} r^2) \Rightarrow O(p^3 b^{\omega-2} r^2)$

$\Rightarrow$ Reduction by a factor $O((b/r)^{3-\omega})$

New Update kernel



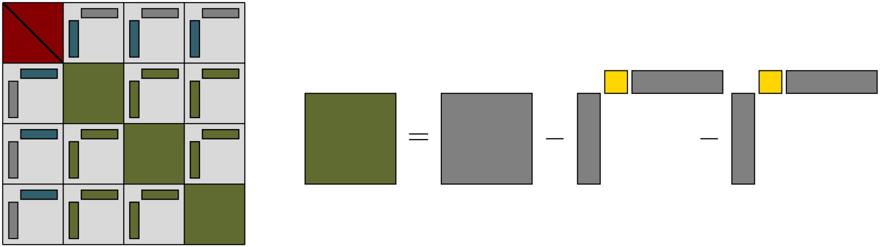


- Classical product: $O(p^3)$ calls of cost $O(br^{\omega-1})$ $\Rightarrow$ $O(p^3br^{\omega-1})$
- New product: $O(p)$ calls of cost $O(p^2b^{\omega-2}r^2)$ $\Rightarrow$ $O(p^3b^{\omega-2}r^2)$

$\Rightarrow$ Reduction by a factor $O((b/r)^{3-\omega})$

- FR target subtractions: $O(p^2b^2r^{\omega-2})$

New Update kernel

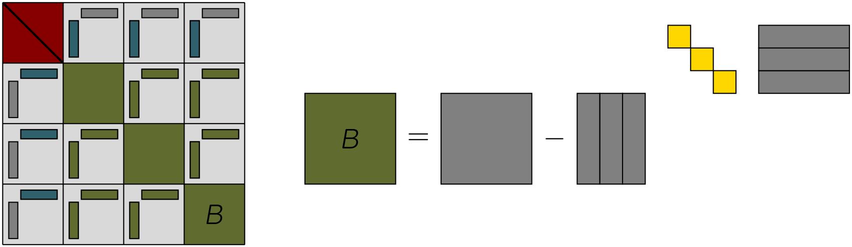


- Classical product: $O(p^3)$ calls of cost $O(br^{\omega-1}) \Rightarrow O(p^3br^{\omega-1})$
- New product: $O(p)$ calls of cost $O(p^2b^{\omega-2}r^2) \Rightarrow O(p^3b^{\omega-2}r^2)$

$\Rightarrow$ Reduction by a factor $O((b/r)^{3-\omega})$

- FR target subtractions: $O(p^2b^2r^{\omega-2})$

New Update kernel



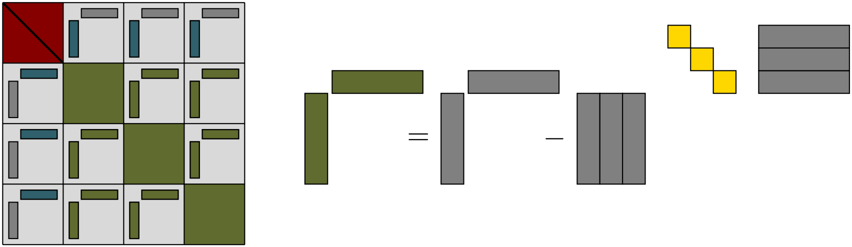
- Classical product: $O(p^3)$ calls of cost $O(br^{\omega-1}) \Rightarrow O(p^3br^{\omega-1})$
- New product: $O(p)$ calls of cost $O(p^2b^{\omega-2}r^2) \Rightarrow O(p^3b^{\omega-2}r^2)$

$\Rightarrow$ Reduction by a factor $O((b/r)^{3-\omega})$

- FR target subtractions: $O(p^2b^2r^{\omega-2}) \rightarrow O(p^2b^{\omega-1}r)$

$\Rightarrow$ Reduction by a factor $O((b/r)^{3-\omega})$

New Update kernel



- Classical product: $O(p^3)$ calls of cost $O(br^{\omega-1}) \Rightarrow O(p^3 br^{\omega-1})$
- New product: $O(p)$ calls of cost $O(p^2 b^{\omega-2} r^2) \Rightarrow O(p^3 b^{\omega-2} r^2)$

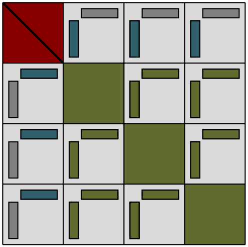
$\Rightarrow$ Reduction by a factor $O((b/r)^{3-\omega})$

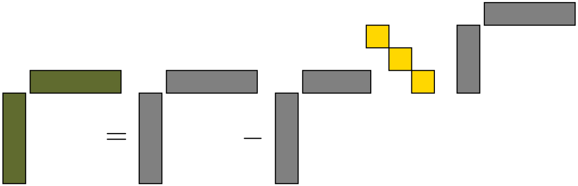
- FR target subtractions: $O(p^2 b^2 r^{\omega-2}) \rightarrow O(p^2 b^{\omega-1} r)$

$\Rightarrow$ Reduction by a factor $O((b/r)^{3-\omega})$

- LR target subtractions: $O(p^3 br^{\omega-1})$

New Update kernel





- Classical product: $O(p^3)$ calls of cost $O(br^{\omega-1}) \Rightarrow O(p^3br^{\omega-1})$
- New product: $O(p)$ calls of cost $O(p^2b^{\omega-2}r^2) \Rightarrow O(p^3b^{\omega-2}r^2)$

$\Rightarrow$ Reduction by a factor $O((b/r)^{3-\omega})$

- FR target subtractions: $O(p^2b^2r^{\omega-2}) \rightarrow O(p^2b^{\omega-1}r)$

$\Rightarrow$ Reduction by a factor $O((b/r)^{3-\omega})$

- LR target subtractions: $O(p^3br^{\omega-1}) \rightarrow O(p^2br^{\omega-1} + p^3r^{\omega})$

$\Rightarrow$ Reduction by a factor $O(b/r) \Rightarrow$ gain even for $\omega = 3$!

Complexity of the new algorithm

- Putting everything together, the complexity of the new algorithm is:
$$O(pb^{\omega} + p^2 b^{\omega-1} r + p^3 b^{\omega-2} r^2)$$

Complexity of the new algorithm

- Putting everything together, the complexity of the new algorithm is:

$$O(pb^{\omega} + p^2 b^{\omega-1} r + p^3 b^{\omega-2} r^2)$$
$$\subset O(n^{(\omega+1)/2} r^{(\omega-1)/2}) \qquad \text{for } b = \Theta((nr)^{1/2})$$

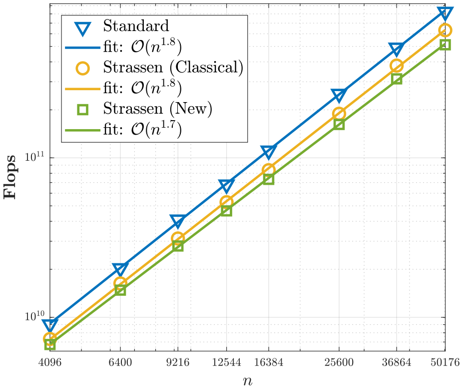
Complexity of the new algorithm

- Putting everything together, the complexity of the new algorithm is:
$$O(pb^{\omega} + p^2 b^{\omega-1} r + p^3 b^{\omega-2} r^2)$$
$$\subset O(n^{(\omega+1)/2} r^{(\omega-1)/2}) \qquad \text{for } b = \Theta((nr)^{1/2})$$

- $\omega = 3 \Rightarrow O(n^2 r)$
- $\omega = 2 \Rightarrow O(n^{3/2} r^{1/2}) \equiv$ BLR storage complexity $\Rightarrow$ nice!
- $\omega = \log_2 7 \Rightarrow O(n^{1.9} r^{0.9})$

Numerical experiments

Experimental validation with a Poisson problem ($r = O(1)$)



	Standard	Strassen (classical)	Strassen (new)
Theory	$O(n^2)$	$O(n^2)$	$O(n^{1.9})$
Experiments	$O(n^{1.8})$	$O(n^{1.8})$	$O(n^{1.7})$

Conclusion

- Block low-rank (BLR) matrices can be factored very efficiently on parallel computers in $O(n^2r)$ flops. We investigated the use of **fast matrix arithmetic** to reduce this complexity

- The classical BLR algorithm is not suited for fast arithmetic and only achieves $O(n^2r^{\omega-2})$ complexity

- We proposed a new algorithm of higher granularity achieving $O(n^{(\omega+1)/2}r^{(\omega-1)/2})$ complexity

- Related work: Pernet & Storjohann show $O(nr^2) \to O(nr^{\omega-1})$ complexity for $\mathcal{H}$ matrices. BLR/$\mathcal{H}$ complexity ratio: $O((n/r)^{(\omega-1)/2}) \Rightarrow$ **fast matrix arithmetic can help bridging the gap between BLR and $\mathcal{H}$ matrices!**

Slides and paper available here

`bit.ly/theomary`

Open questions/perspectives

1. **High performance implementation:** 40% reduction in flops for $n \sim 50,000 \Rightarrow$ how much can be converted in effective time gains? Which fast algorithms will be practical? (Strassen?)

   New algorithm could be beneficial even outside the context of fast arithmetic (e.g., for GPU architectures)

2. **Fast numerical rank revealing factorization (NRRF):** our complexity analysis requires a NRRF of cost $O(mnr^{\omega-2})$ $\Rightarrow$ actually not straightforward, no papers on this topic?

   Classical NRRF cannot efficiently exploit fast arithmetic, e.g., truncated CPQR requires $O(mnr)$ BLAS-2 flops. Randomized approaches could be a solution?

   1. $S \leftarrow A\Omega \quad \rightarrow O(mnr^{\omega-2})$
   2. $Q \leftarrow \mathtt{qr}(S) \quad \rightarrow O(mr^{\omega-1})$
   3. $Y \leftarrow A^T Q \quad \rightarrow O(mnr^{\omega-2})$

   ($A \approx QQ^T A$ and thus QY^T is a LR approximation of A)