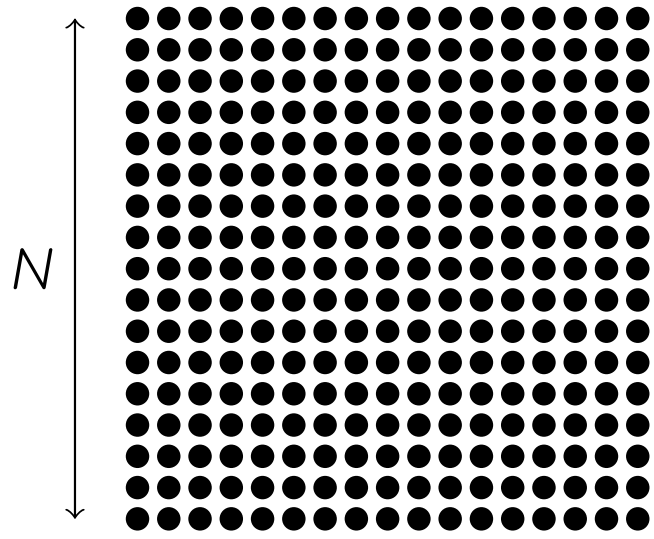
Improving multifrontal solvers by means of Block Low-Rank approximations

The MUMPS team, INP-IRIT, INRIA-LIP, Université de Bordeaux, CNRS-IRIT

Journée problème de Poisson, IHP Paris 26/01/2015

The Multifrontal method

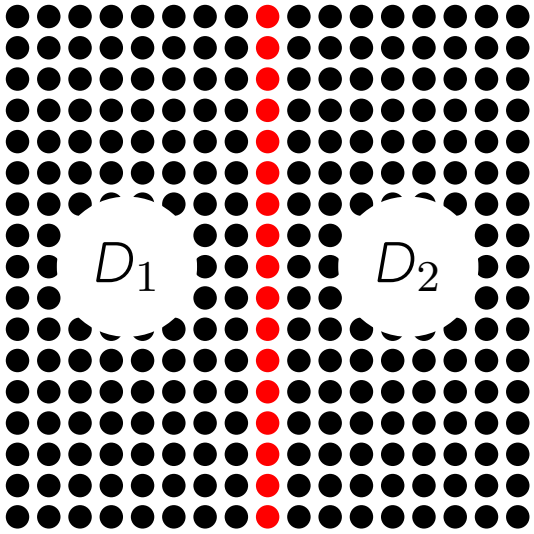
MF (Duff'83) ND (George'73)

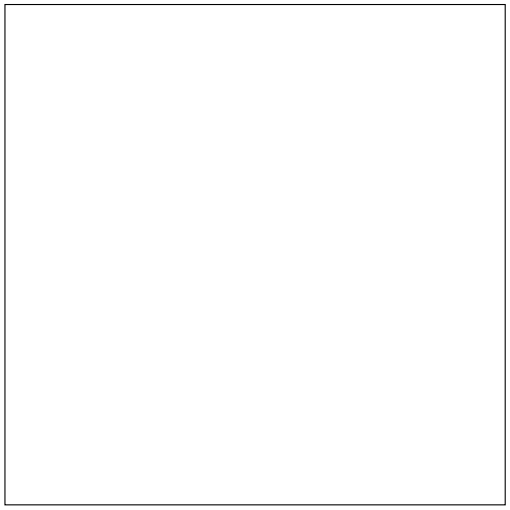


2D problem cost $\propto$

Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$

MF (Duff'83) ND (George'73)

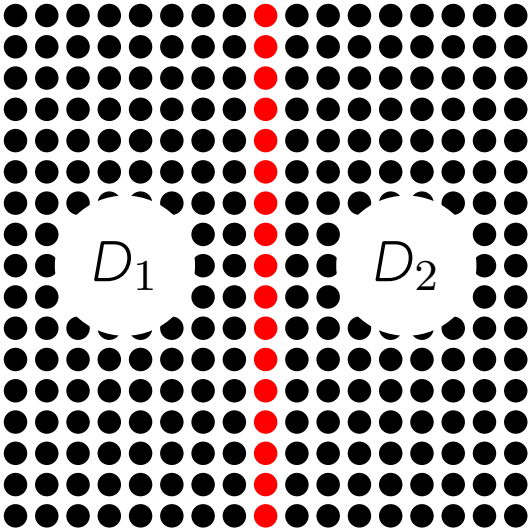


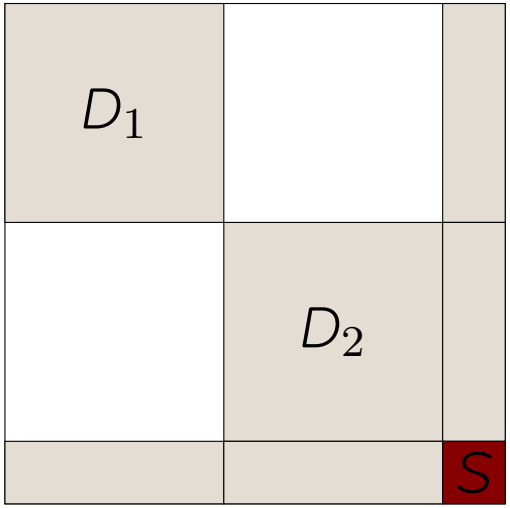


2D problem cost $\propto$

Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$

MF (Duff'83) ND (George'73)

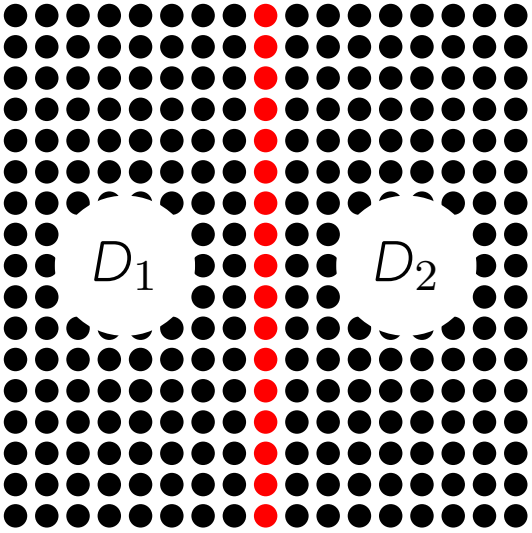


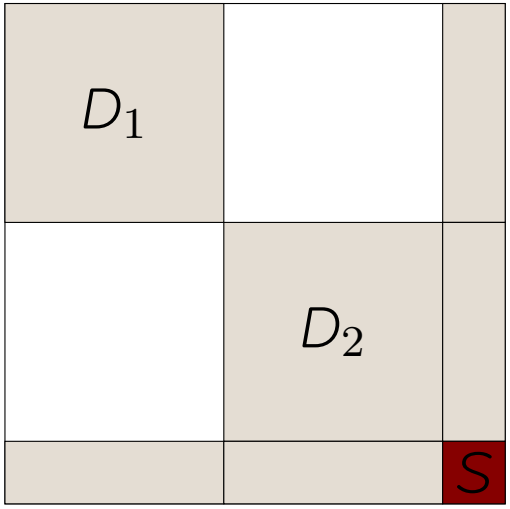


2D problem cost $\propto$

Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$

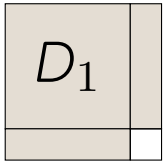
MF (Duff'83) ND (George'73)



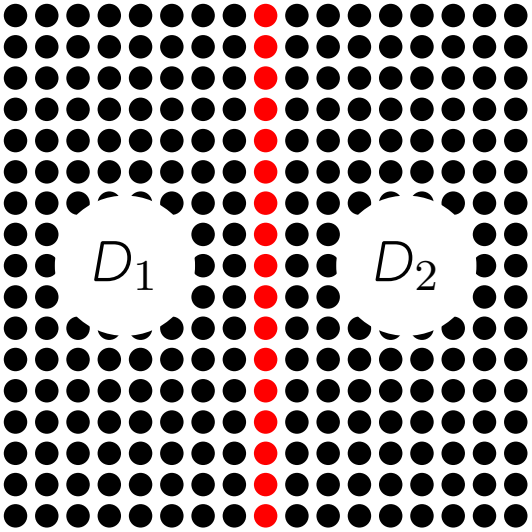


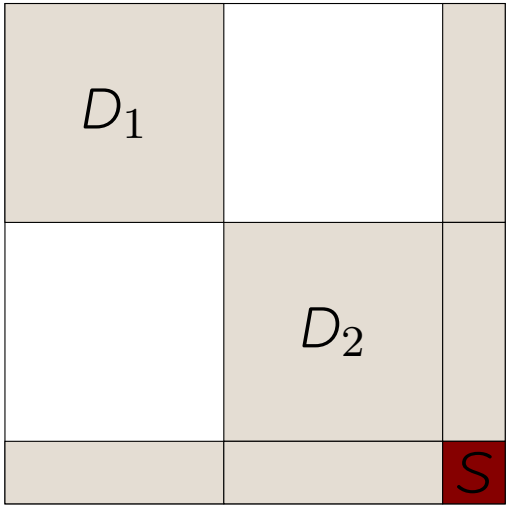
2D problem cost $\propto$

Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$



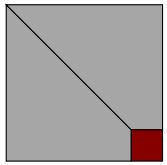
MF (Duff'83) ND (George'73)



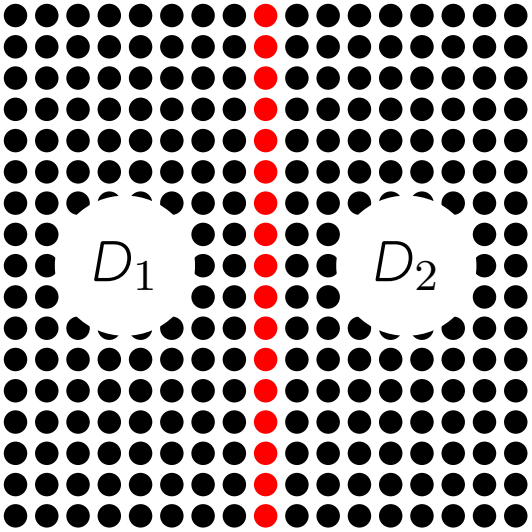


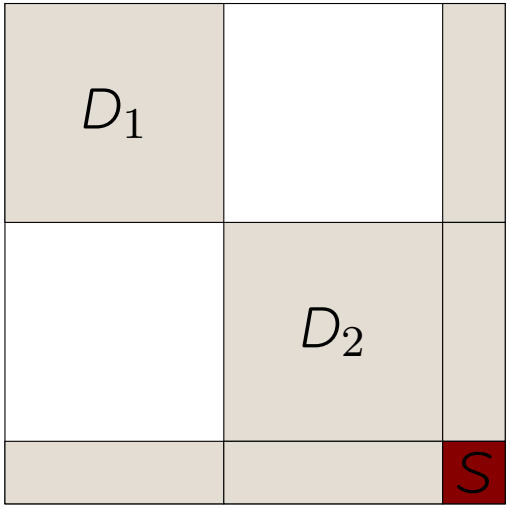
2D problem cost $\propto$

Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$



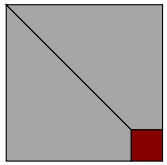
MF (Duff'83) ND (George'73)

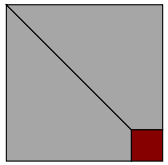




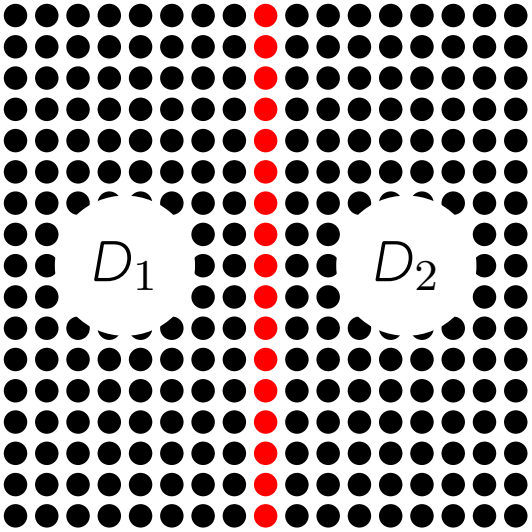
2D problem cost $\propto$

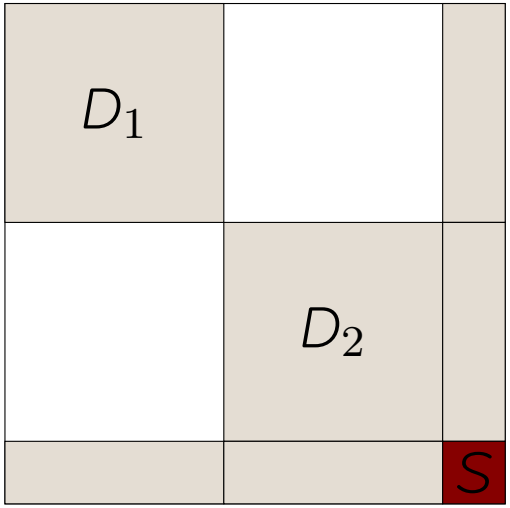
Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$





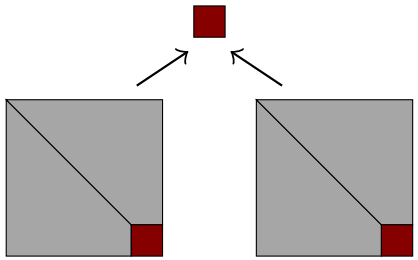
MF (Duff'83) ND (George'73)



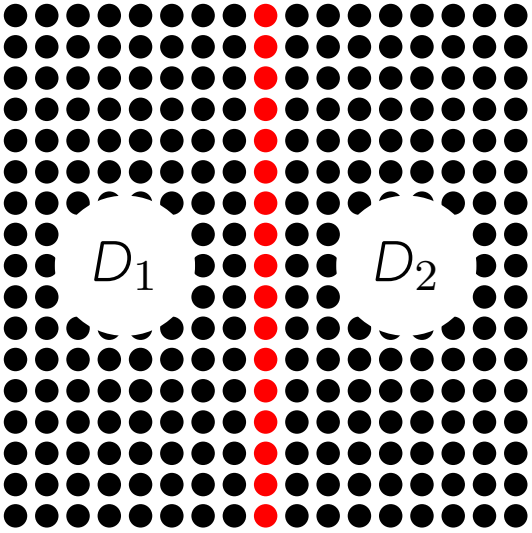


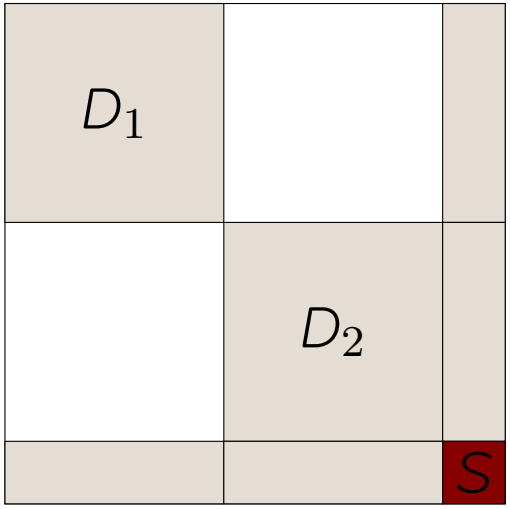
2D problem cost $\propto$

Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$



MF (Duff'83) ND (George'73)

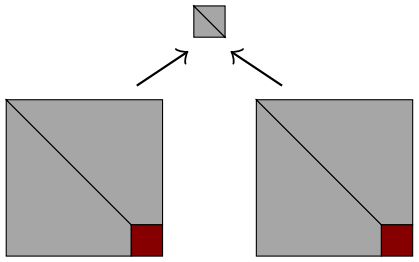




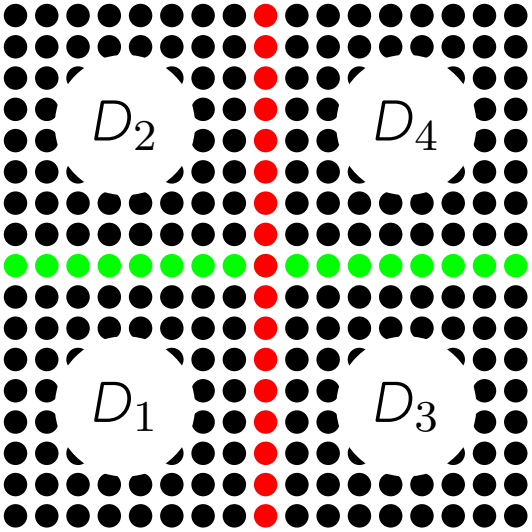
2D problem cost $\propto$

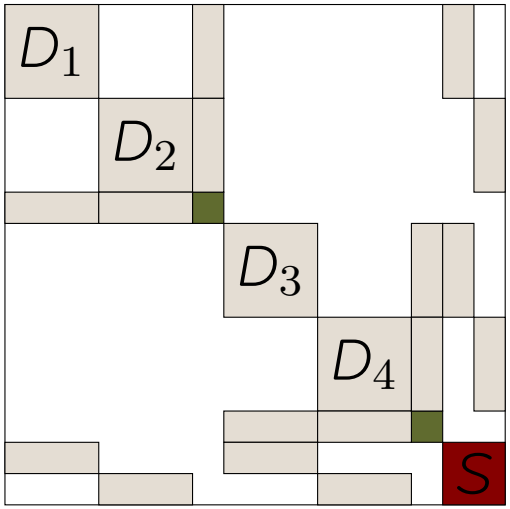
Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$

$\rightarrow$ Flops:$\mathcal{O}(N^6/8)$, mem:$\mathcal{O}(N^4/2)$



MF (Duff'83) ND (George'73)





2D problem cost $\propto$

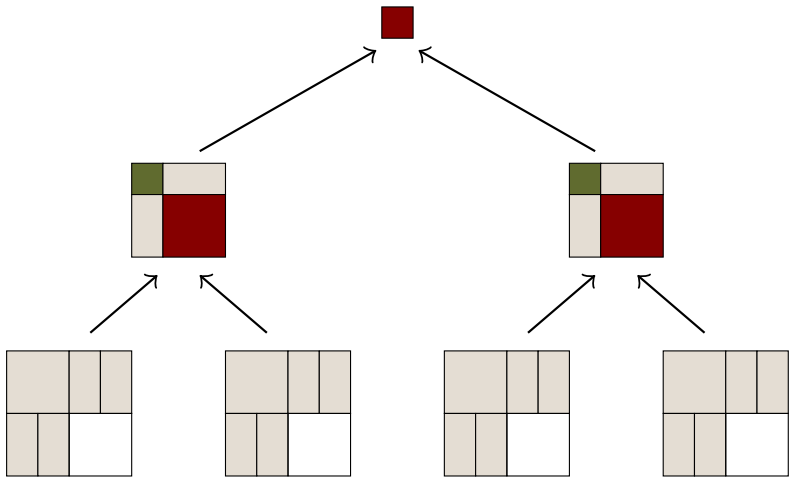
Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$

$\rightarrow$ Flops:$\mathcal{O}(N^6/8)$, mem:$\mathcal{O}(N^4/2)$

$\rightarrow$ Flops:$\mathcal{O}(N^3)$, mem:$\mathcal{O}(N^2 log(N))$

3D problem cost $\propto$

$\rightarrow$ Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$



The Multifrontal method

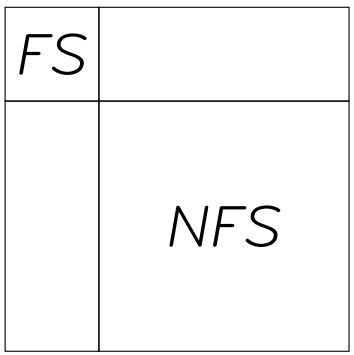
Important things to remember about MF:

- the elimination tree can be traversed in any topological order
- two sources of parallelism:
  1. Tree: concurrent processing for nodes in different branches
  2. Node: parallel processing for big nodes
- many small nodes at the bottom, few but large on top
- two types of variables in each front: Fully Summed (FS) and Non-FS
- delayed pivoting is a necessary evil.

The Multifrontal method

Important things to remember about MF:

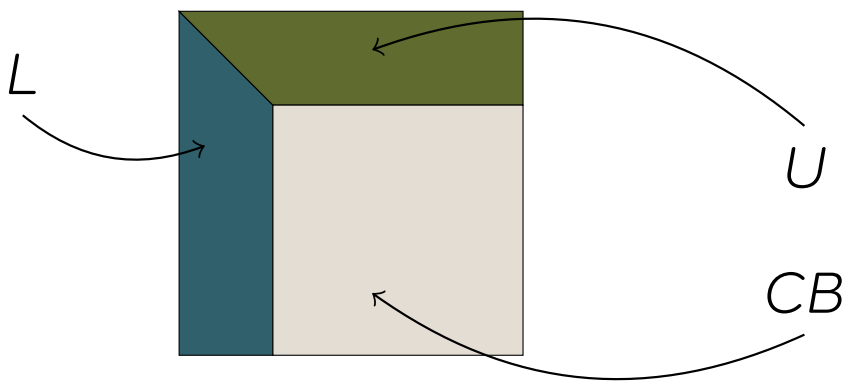
- the elimination tree can be traversed in any topological order
- two sources of parallelism:
  1. Tree: concurrent processing for nodes in different branches
  2. Node: parallel processing for big nodes
- many small nodes at the bottom, few but large on top
- two types of variables in each front: Fully Summed (FS) and Non-FS
- delayed pivoting is a necessary evil.



The Multifrontal method

Important things to remember about MF:

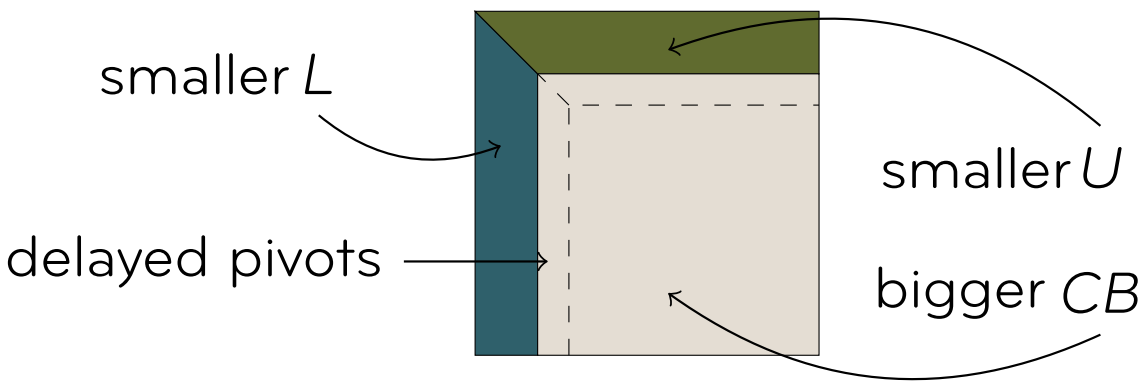
- the elimination tree can be traversed in any topological order
- two sources of parallelism:
  1. Tree: concurrent processing for nodes in different branches
  2. Node: parallel processing for big nodes
- many small nodes at the bottom, few but large on top
- two types of variables in each front: Fully Summed (FS) and Non-FS
- delayed pivoting is a necessary evil.



The Multifrontal method

Important things to remember about MF:

- the elimination tree can be traversed in any topological order
- two sources of parallelism:
  1. Tree: concurrent processing for nodes in different branches
  2. Node: parallel processing for big nodes
- many small nodes at the bottom, few but large on top
- two types of variables in each front: Fully Summed (FS) and Non-FS
- delayed pivoting is a necessary evil. If a pivot does not match a stability criterion, its elimination is postponed to the parent front



The Multifrontal method

Advantages over iterative solvers:

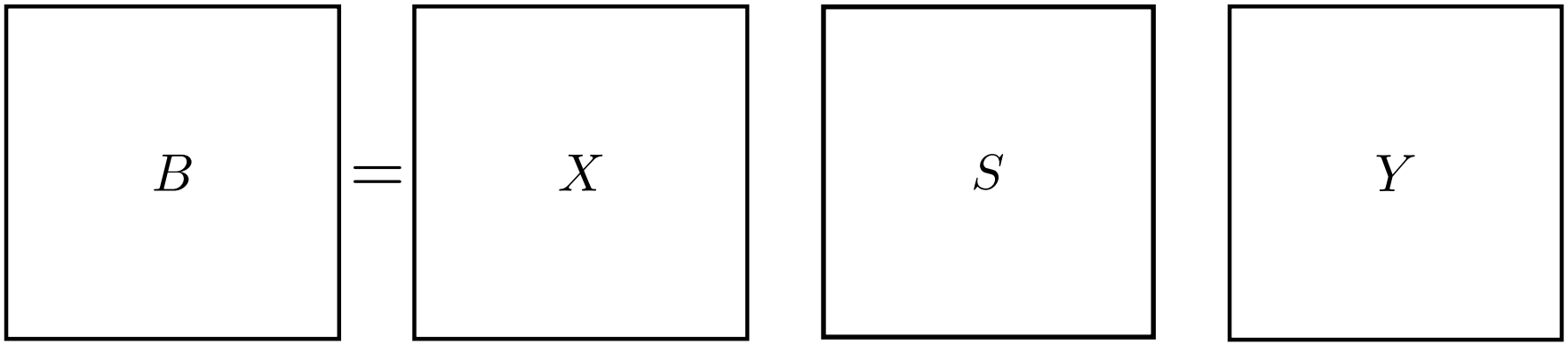
- easy to use (push button $\rightarrow$ get answer)
- numerically robust
- do one factorization and multiple bw/fw substitutions
- direct solvers are Swiss army knives:
  - solve system
  - compute Schur complement
  - compute rank/null-space
  - compute (selected entries of) the inverse matrix
  - ...
- can be used to precondition iterative solvers

All these features come at the price of high memory and CPU consumption. Low-rank approximations can help.

Block Low-Rank representations

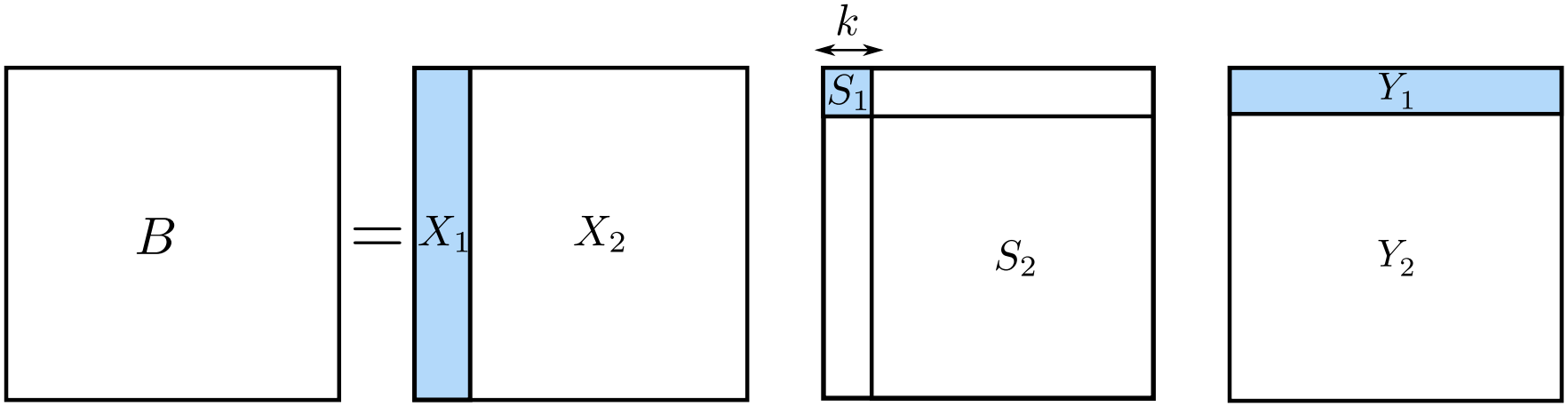
Low-rank matrices

Take a dense matrix B of size $n \times n$ and compute its SVD $B = XSY$:



Low-rank matrices

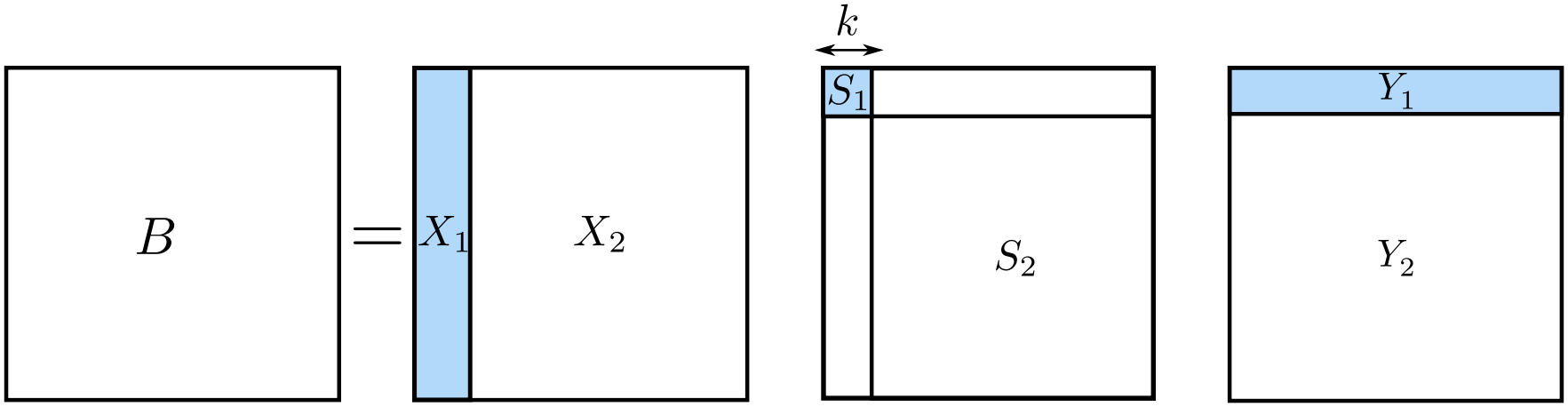
Take a dense matrix B of size $n \times n$ and compute its SVD $B = XSY$:



$$B = X_1 S_1 Y_1 + X_2 S_2 Y_2 \quad \text{with} \quad S_1(k,k) = \sigma_k > \varepsilon,\ S_2(1,1) = \sigma_{k+1} \leq \varepsilon$$

Low-rank matrices

Take a dense matrix B of size $n \times n$ and compute its SVD $B = XSY$:

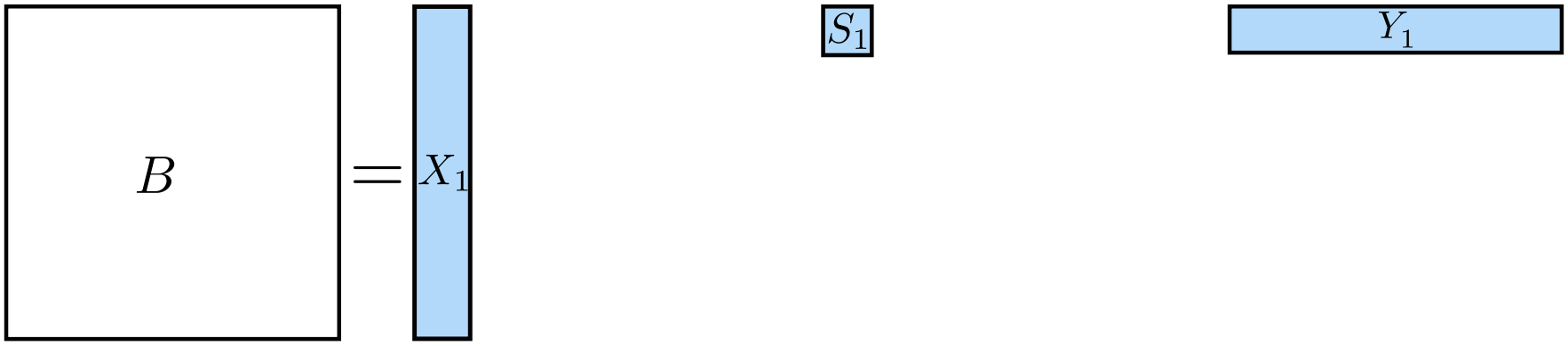


$$B = X_1 S_1 Y_1 + X_2 S_2 Y_2 \quad \text{with} \quad S_1(k,k) = \sigma_k > \varepsilon,\ S_2(1,1) = \sigma_{k+1} \leq \varepsilon$$

If $\tilde{B} = X_1 S_1 Y_1$ then $\|B - \tilde{B}\|_2 = \|X_2 S_2 Y_2\|_2 = \sigma_{k+1} \leq \varepsilon$

Low-rank matrices

Take a dense matrix B of size $n \times n$ and compute its SVD $B = XSY$:



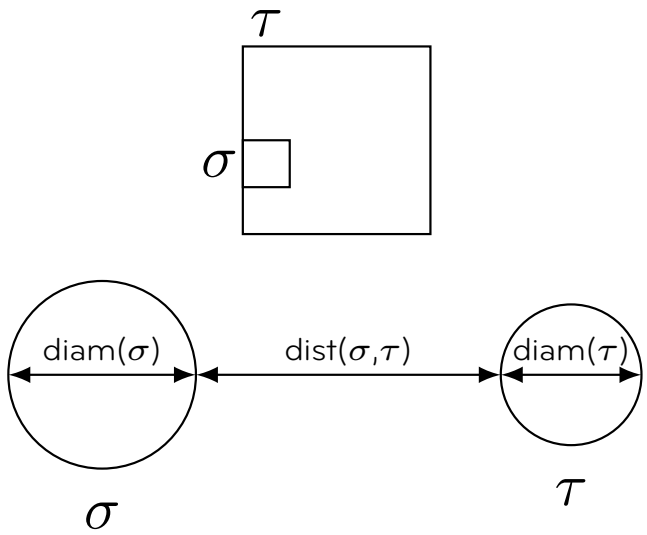
$B = X_1 S_1 Y_1 + X_2 S_2 Y_2$ with $S_1(k,k) = \sigma_k > \varepsilon,\ S_2(1,1) = \sigma_{k+1} \leq \varepsilon$

If $\tilde{B} = X_1 S_1 Y_1$ then $\|B - \tilde{B}\|_2 = \|X_2 S_2 Y_2\|_2 = \sigma_{k+1} \leq \varepsilon$

If the singular values of B decay very fast (e.g. exponentially) then $k \ll n$ even for very small ε (e.g. 10^{-14}) $\Rightarrow$ memory and CPU consumption can be reduced considerably with a controlled loss of accuracy ($\leq \varepsilon$) if $\tilde{B}$ is used instead of B

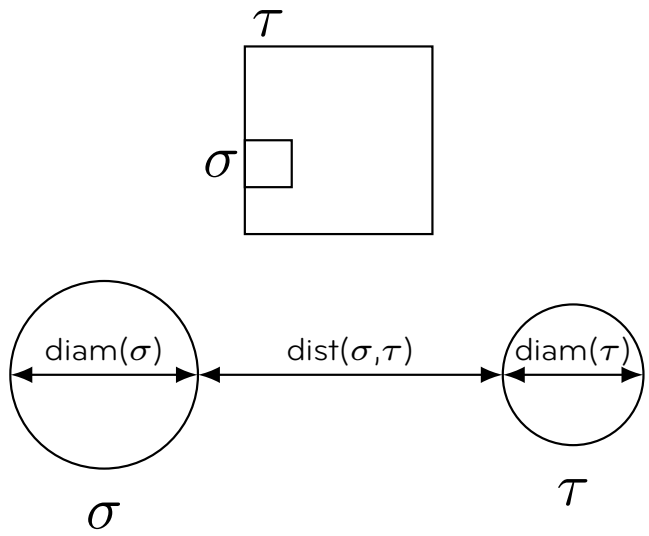
Can we exploit low-rankness in general dense matrices?

Dense matrices are usually not low-rank but in many applications they exhibit low-rank blocks.
A block represents the interaction between two subdomains σ and τ. If they are small and far away the interaction is weak $\Rightarrow$ rank is low



Can we exploit low-rankness in general dense matrices?

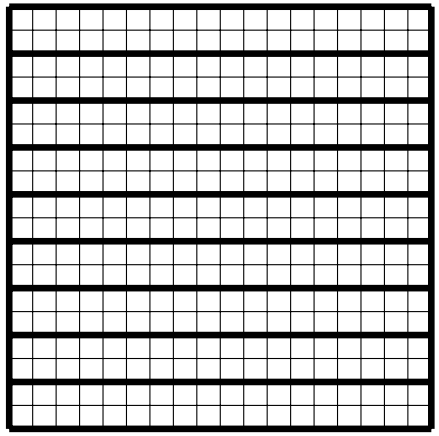
Dense matrices are usually not low-rank but in many applications they exhibit low-rank blocks.
A block represents the interaction between two subdomains σ and τ. If they are small and far away the interaction is weak $\Rightarrow$ rank is low



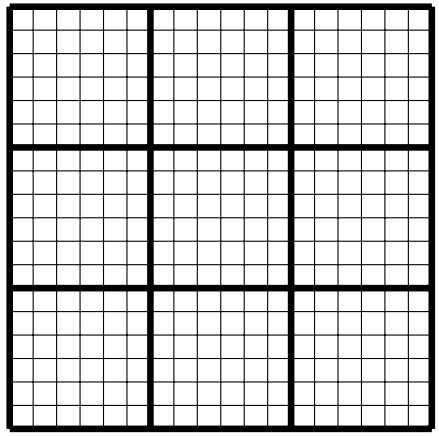
1. compute a clustering of your domain (mesh)
2. permute the matrix accordingly
3. enjoy low-rankness

Clustering

Clustering with as many low-rank blocks as possible



large diameters
small distances

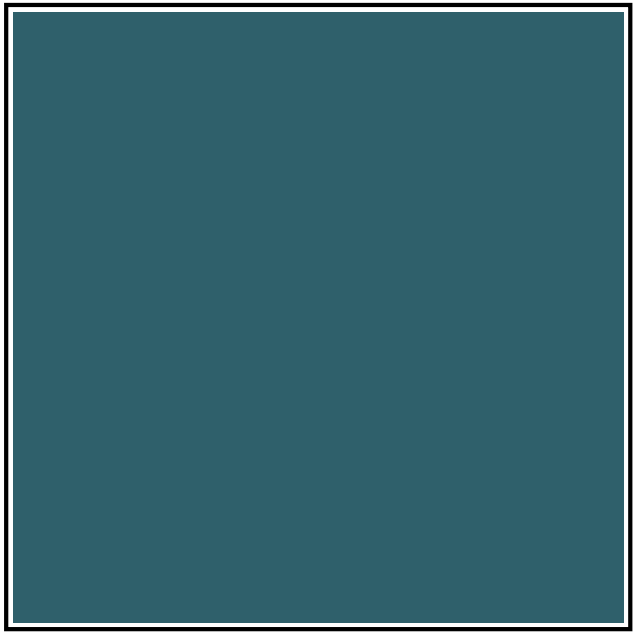


small diameters
large distances

Any partitioning tool like Metis or SCOTCH will get you a good clustering of your domain

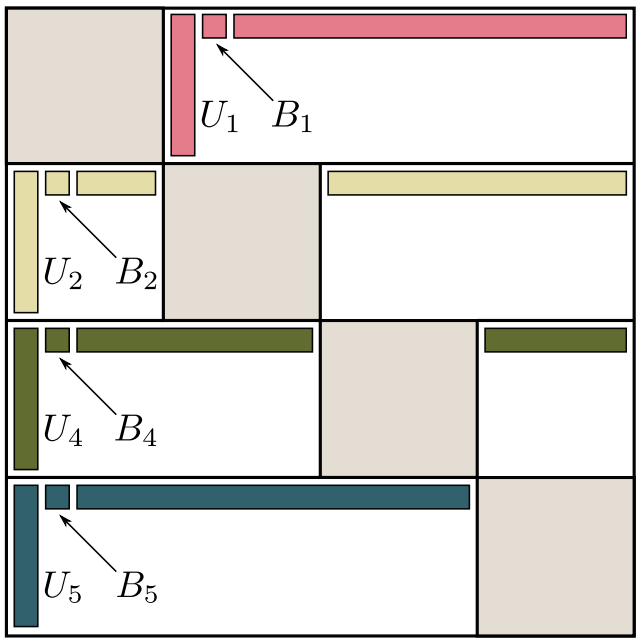
Low-rank approximations – representations

Once the blocking is defined, several low-rank formats are possible. One is Hierarchically Semi-Separable (HSS)"



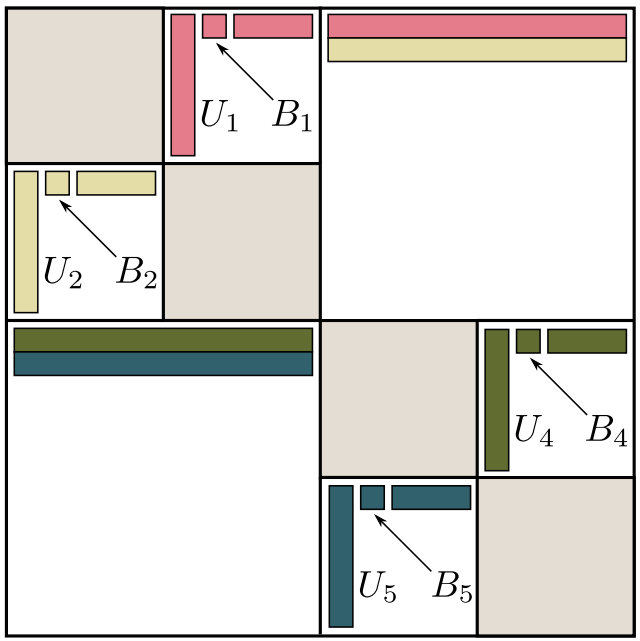
Low-rank approximations – representations

Once the blocking is defined, several low-rank formats are possible. One is Hierarchically Semi-Separable (HSS)"



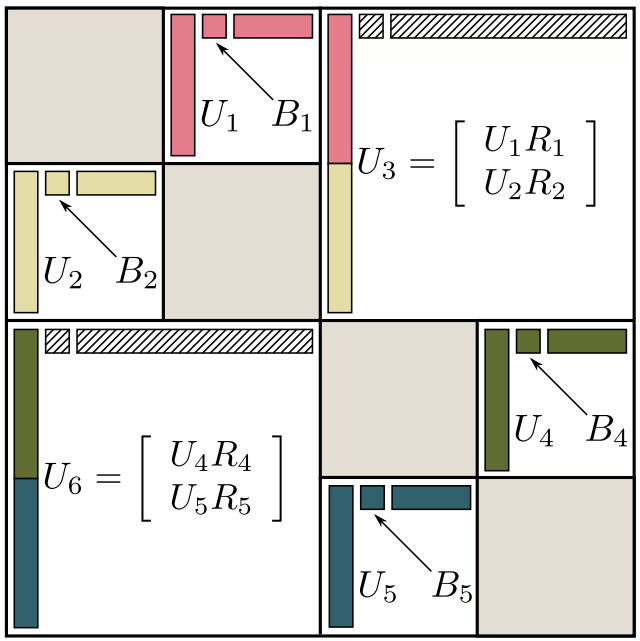
Low-rank approximations – representations

Once the blocking is defined, several low-rank formats are possible. One is Hierarchically Semi-Separable (HSS)"



Low-rank approximations – representations

Once the blocking is defined, several low-rank formats are possible. One is Hierarchically Semi-Separable (HSS)"

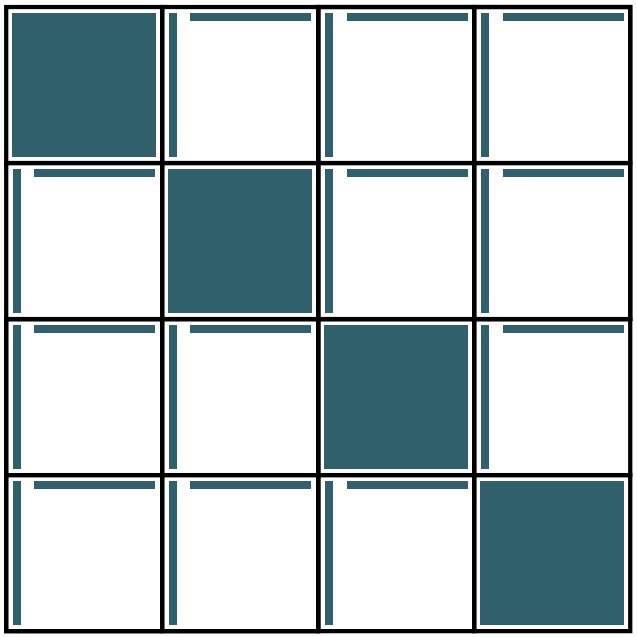


HSS:

- Leads to very low complexity (fact. is $O(n)$, with a big constant)
- complex, hierarchical structure
- relatively inefficient and expensive SVD/RRQR... (very T&S blocks)
- parallelism is difficult to exploit

Low-rank approximations – representations

Once the blocking is defined, several low-rank formats are possible. One is Hierarchically Semi-Separable (HSS)"

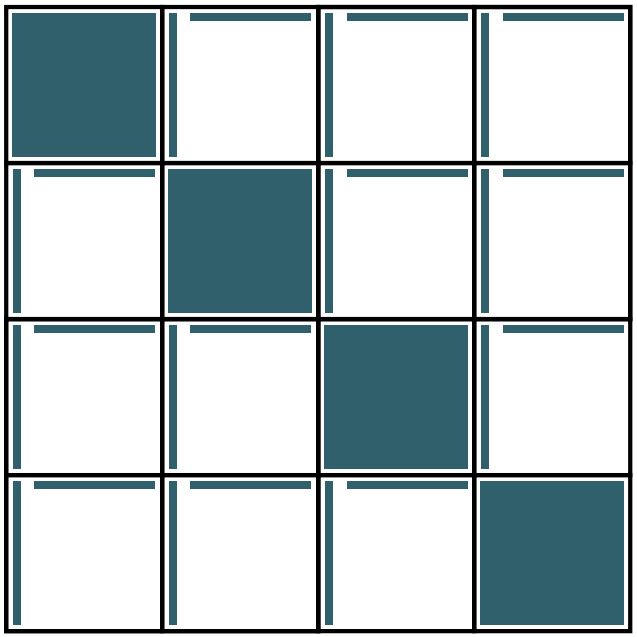


BLR:

- Higher complexity
- Very simple structure (very little logic to handle)
- cheap SVD/RRQR
- completely parallel

Low-rank approximations – representations

Once the blocking is defined, several low-rank formats are possible. One is Hierarchically Semi-Separable (HSS)"



BLR:

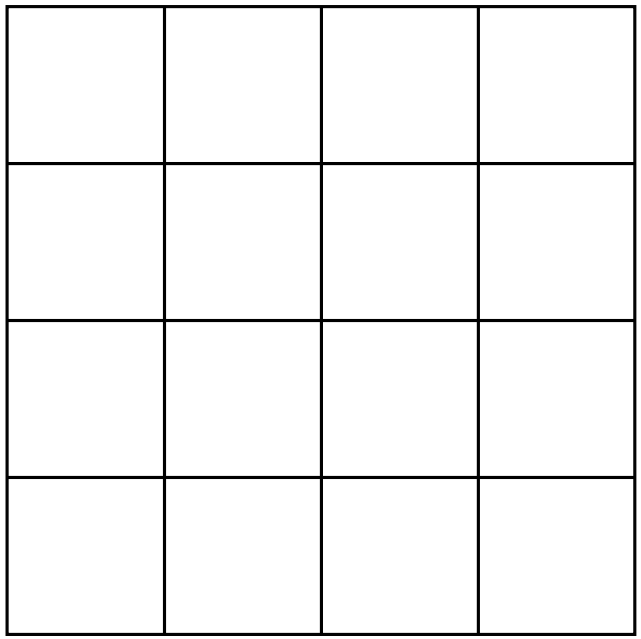
- Higher complexity
- Very simple structure (very little logic to handle)
- cheap SVD/RRQR
- completely parallel

We believe Block Low-Rank (BLR) provides a good compromise between complexity and performance/usability

Dense BLR LU factorization

task	operation type	dense	low-rank
Compress (C)	$B = XY$	kr^2	–
Factor (F)	$B = LU^T$	$(2/3)r^3$	$(2/3)r^3$
Solve (S)	$B = X(YL^{-1})$	r^3	kr^2
Update (U)	$B = B - X_1(Y_1X_2)Y_2$	$2r^3$	$2kr^2$

(r=block size, k=rank)



_GETRF

_TRSM

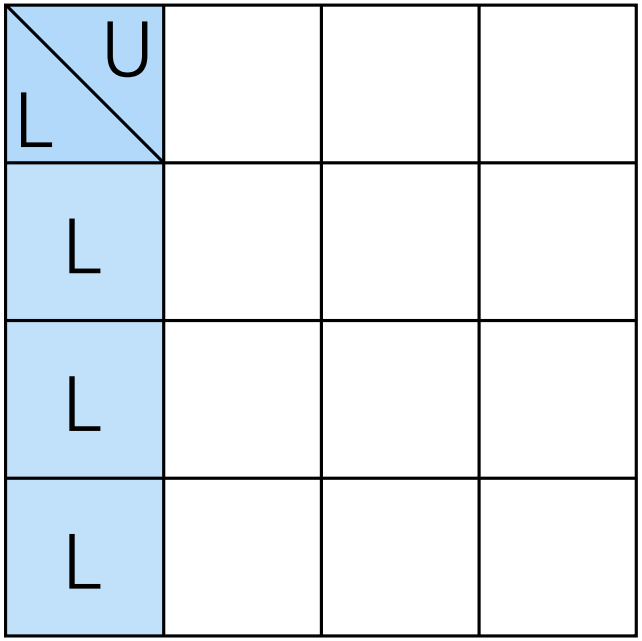
_GEQP3/_GESVD

_GEMM

Dense BLR LU factorization

task	operation type	dense	low-rank
Compress (C)	$B = XY$	kr^2	–
Factor (F)	$B = LU^T$	$(2/3)r^3$	$(2/3)r^3$
Solve (S)	$B = X(YL^{-1})$	r^3	kr^2
Update (U)	$B = B - X_1(Y_1X_2)Y_2$	$2r^3$	$2kr^2$

(r=block size, k=rank)

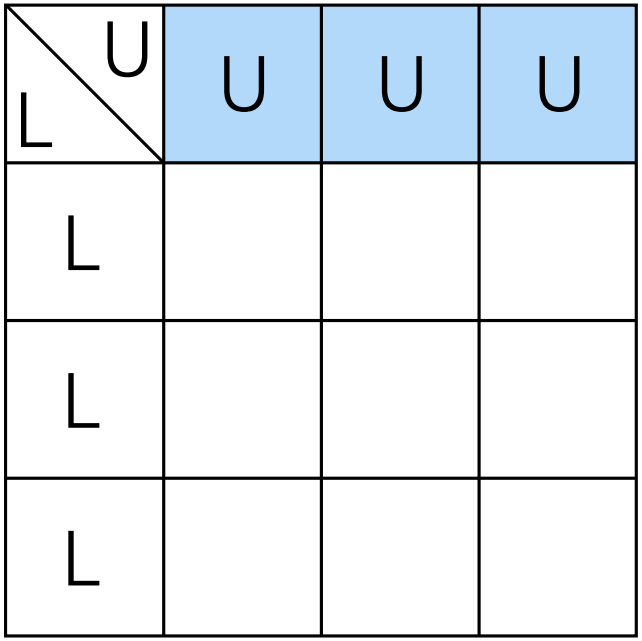


- ▶ _GETRF
- _TRSM
- _GEQP3/_GESVD
- _GEMM

Dense BLR LU factorization

task	operation type	dense	low-rank
Compress (C)	$B = XY$	kr^2	–
Factor (F)	$B = LU^T$	$(2/3)r^3$	$(2/3)r^3$
Solve (S)	$B = X(YL^{-1})$	r^3	kr^2
Update (U)	$B = B - X_1(Y_1X_2)Y_2$	$2r^3$	$2kr^2$

(r=block size, k=rank)

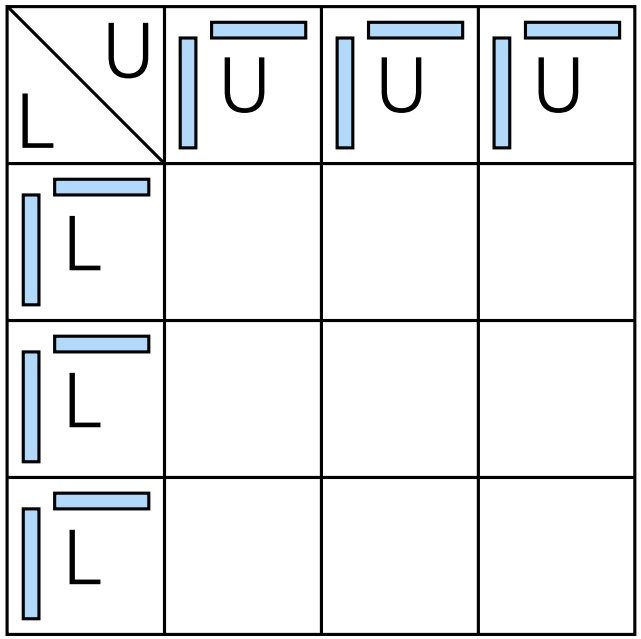


- _GETRF
- ▶ _TRSM
- _GEQP3/_GESVD
- _GEMM

Dense BLR LU factorization

task	operation type	dense	low-rank
Compress (C)	$B = XY$	kr^2	–
Factor (F)	$B = LU^T$	$(2/3)r^3$	$(2/3)r^3$
Solve (S)	$B = X(YL^{-1})$	r^3	kr^2
Update (U)	$B = B - X_1(Y_1X_2)Y_2$	$2r^3$	$2kr^2$

(r=block size, k=rank)

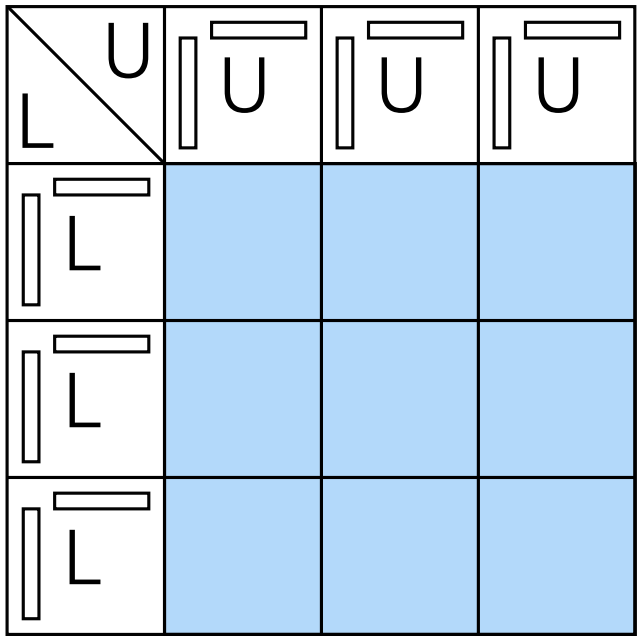


- _GETRF
- _TRSM
- ▶ _GEQP3/_GESVD
- _GEMM

Dense BLR LU factorization

task	operation type	dense	low-rank
Compress (C)	$B = XY$	kr^2	–
Factor (F)	$B = LU^T$	$(2/3)r^3$	$(2/3)r^3$
Solve (S)	$B = X(YL^{-1})$	r^3	kr^2
Update (U)	$B = B - X_1(Y_1X_2)Y_2$	$2r^3$	$2kr^2$

(r=block size, k=rank)

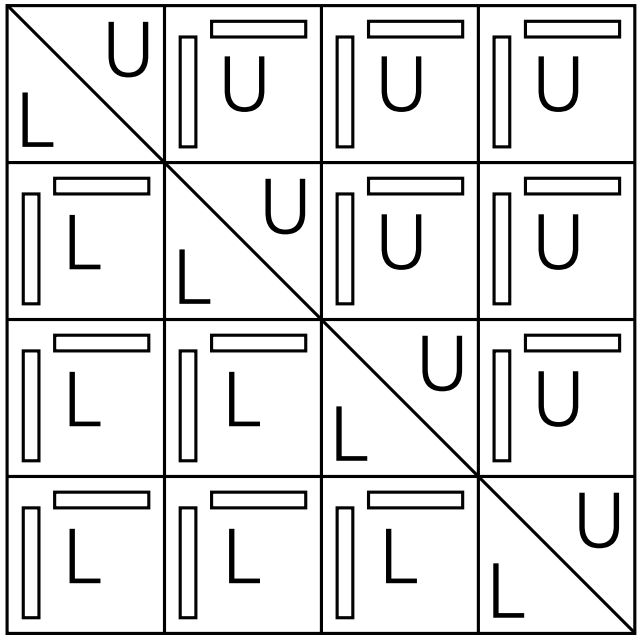


- _GETRF
- _TRSM
- _GEQP3/_GESVD
- ▶ _GEMM

Dense BLR LU factorization

task	operation type	dense	low-rank
Compress (C)	$B = XY$	kr^2	–
Factor (F)	$B = LU^T$	$(2/3)r^3$	$(2/3)r^3$
Solve (S)	$B = X(YL^{-1})$	r^3	kr^2
Update (U)	$B = B - X_1(Y_1X_2)Y_2$	$2r^3$	$2kr^2$

(r=block size, k=rank)



_GETRF

_TRSM

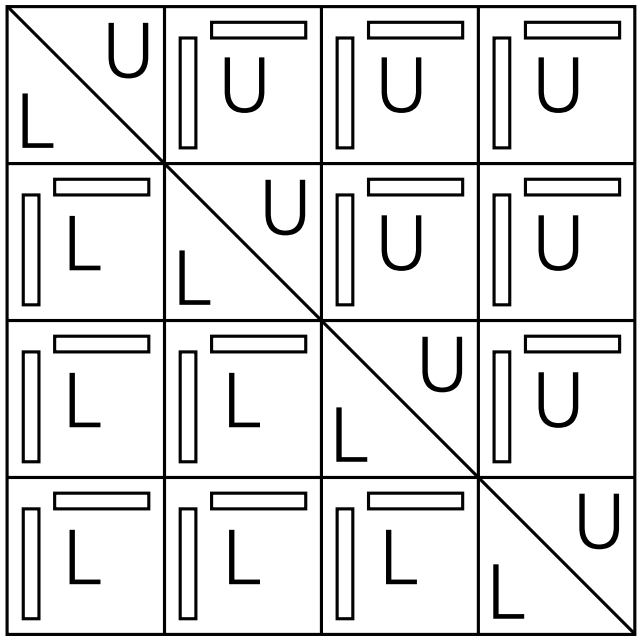
_GEQP3/_GESVD

_GEMM

Dense BLR LU factorization

task	operation type	dense	low-rank
Compress (C)	$B = XY$	kr^2	–
Factor (F)	$B = LU^T$	$(2/3)r^3$	$(2/3)r^3$
Solve (S)	$B = X(YL^{-1})$	r^3	kr^2
Update (U)	$B = B - X_1(Y_1X_2)Y_2$	$2r^3$	$2kr^2$

(r=block size, k=rank)



_GETRF

_TRSM

_GEQP3/_GESVD

_GEMM

other variants possible with different objectives

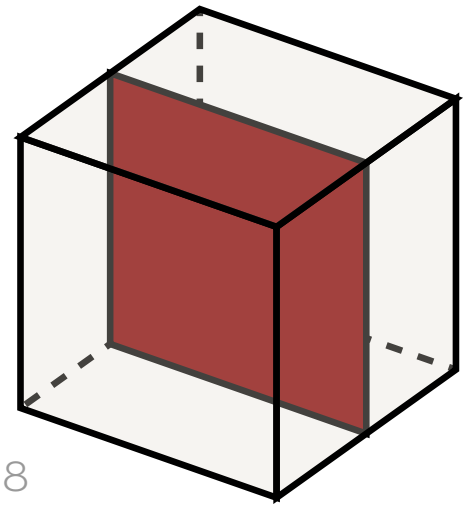
Experimental dense complexity

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

$$\Delta u = f$$

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 $s(x, \omega)$.

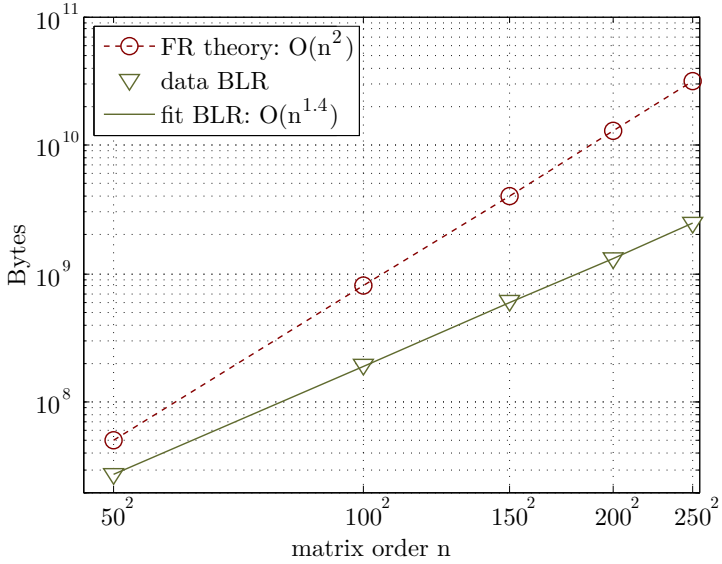
$$\left(-\Delta - \frac{\omega^2}{v(x)^2}\right) u(x, \omega) = s(x, \omega)$$



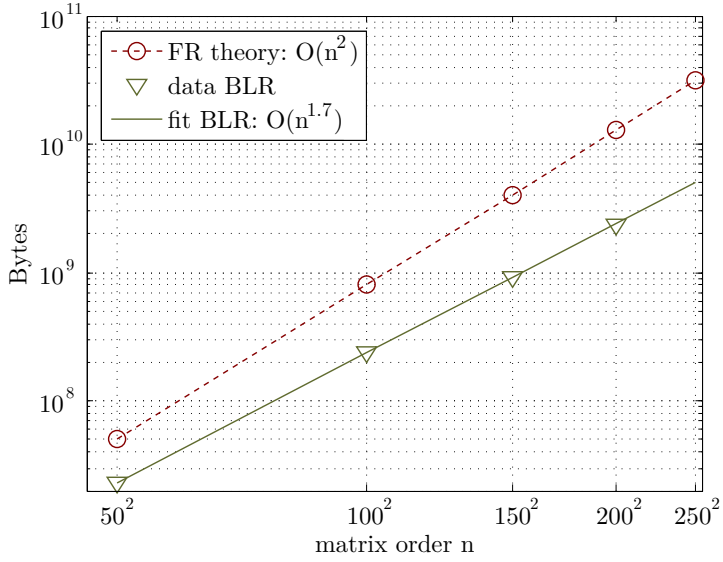
Use the dense matrix defined as the Schur complement associated with the topmost separator

Experimental dense complexity: entries in factor

Poisson ($\varepsilon = 10^{-14}$) ($n = N^2$) Helmholtz ($\varepsilon = 10^{-8}$)



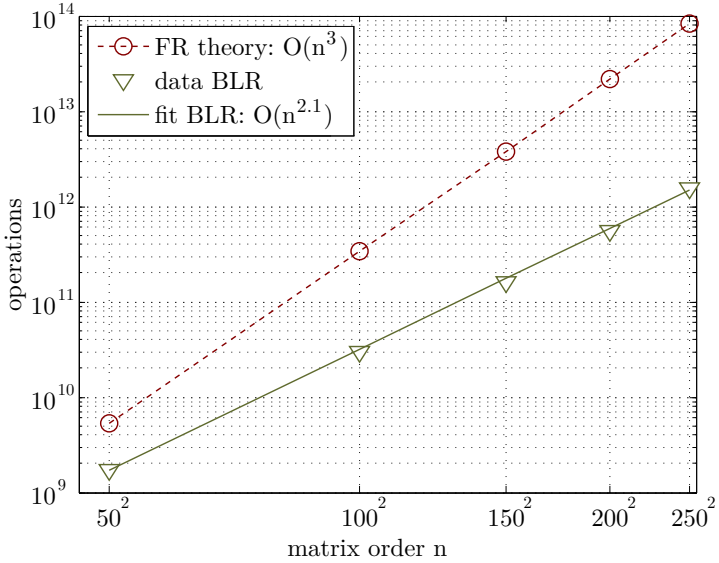
N= 250 $\Rightarrow$ almost 15 times fewer entries



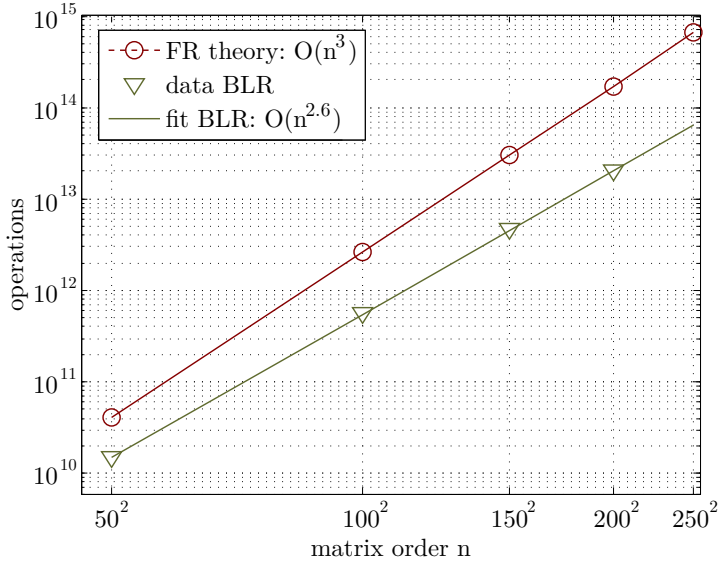
N= 200 $\Rightarrow$ almost 10 times fewer entries

Experimental dense complexity: ops for facto

Poisson ($\varepsilon = 10^{-14}$) ($n = N^2$) Helmholtz ($\varepsilon = 10^{-8}$)



N= 250 $\Rightarrow$ almost 100 times less flops

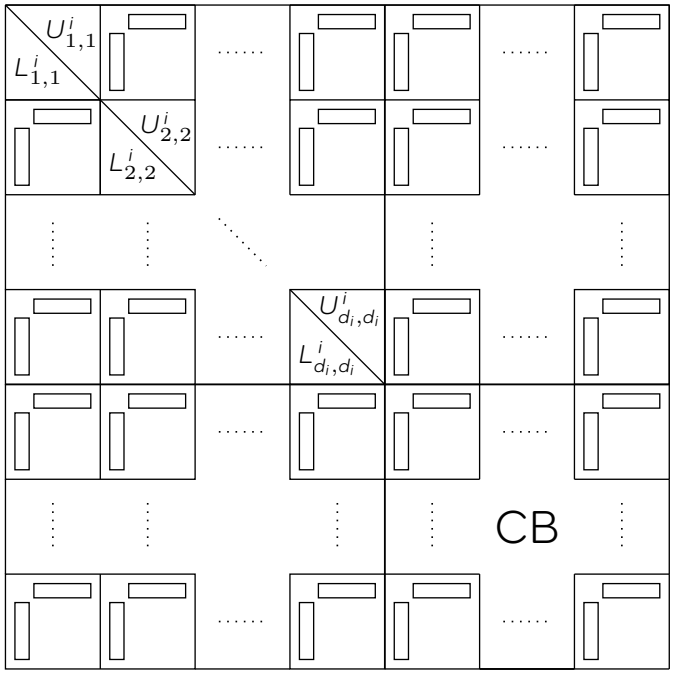


N= 200 $\Rightarrow$ almost 10 times less flops

Block Low-Rank in the multifrontal method

Principle

Fronts are not low-rank but have low-rank subblocks



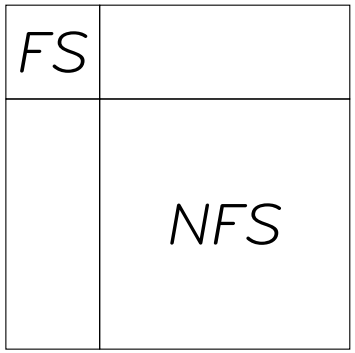
$\Rightarrow$ dense algorithm (FSCU) adapts to partial factorization

$\Rightarrow$ only the largest fronts are processed with BLR

Algebraic clustering/blocking

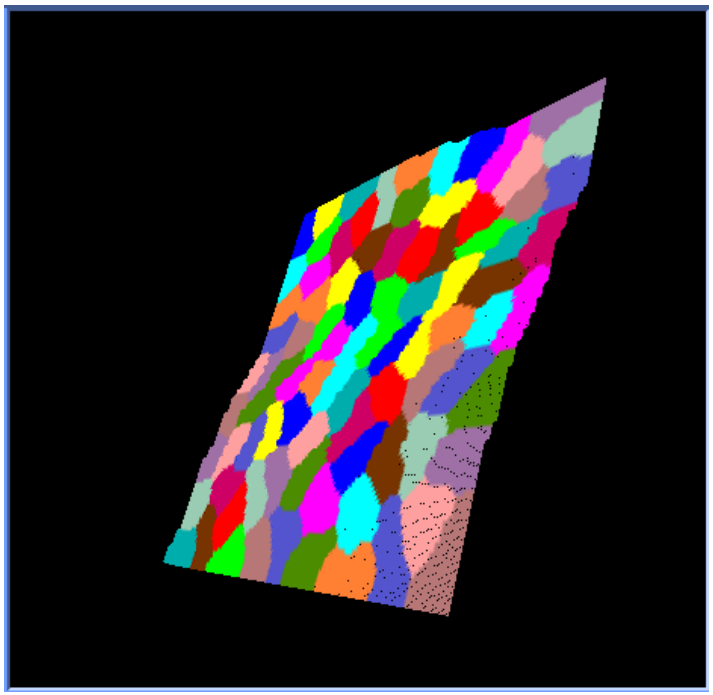
All the variables in a front belong to a separator

- FS: to the separator associated with the front
- NFS: to separator associated with ancestors

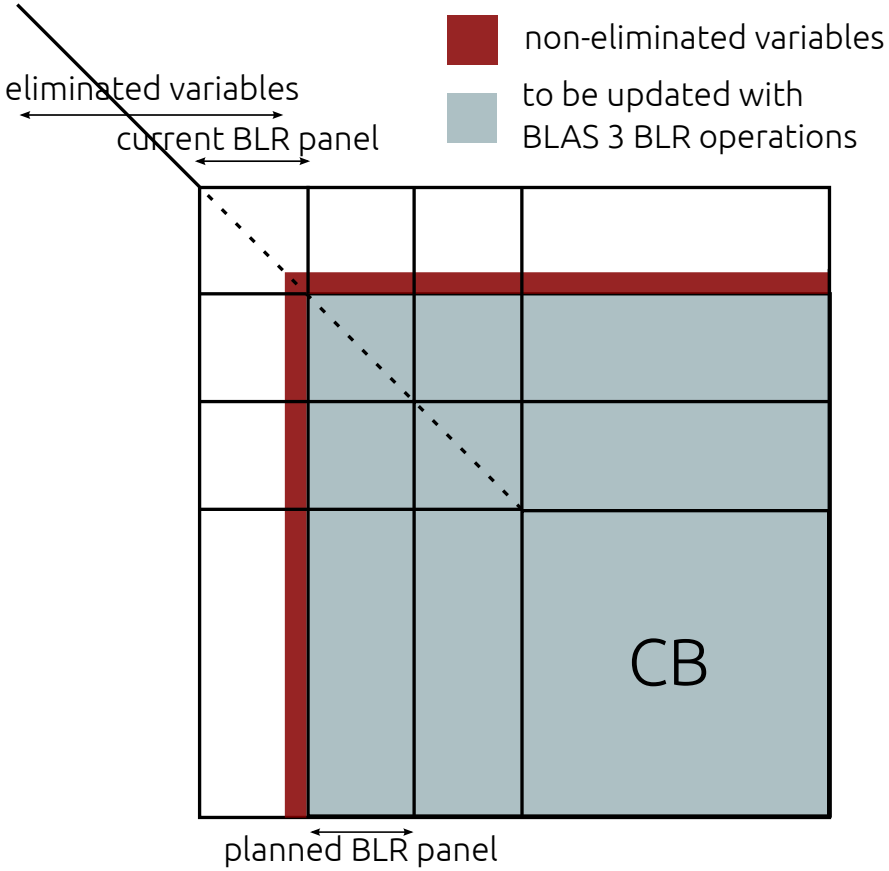


Clustering is achieved by running a partitioning tool on the adjacency subgraphs associated with each separator

SCOTCH-partitioned SCOTCH separator on a cubic domain of size $N = 128$

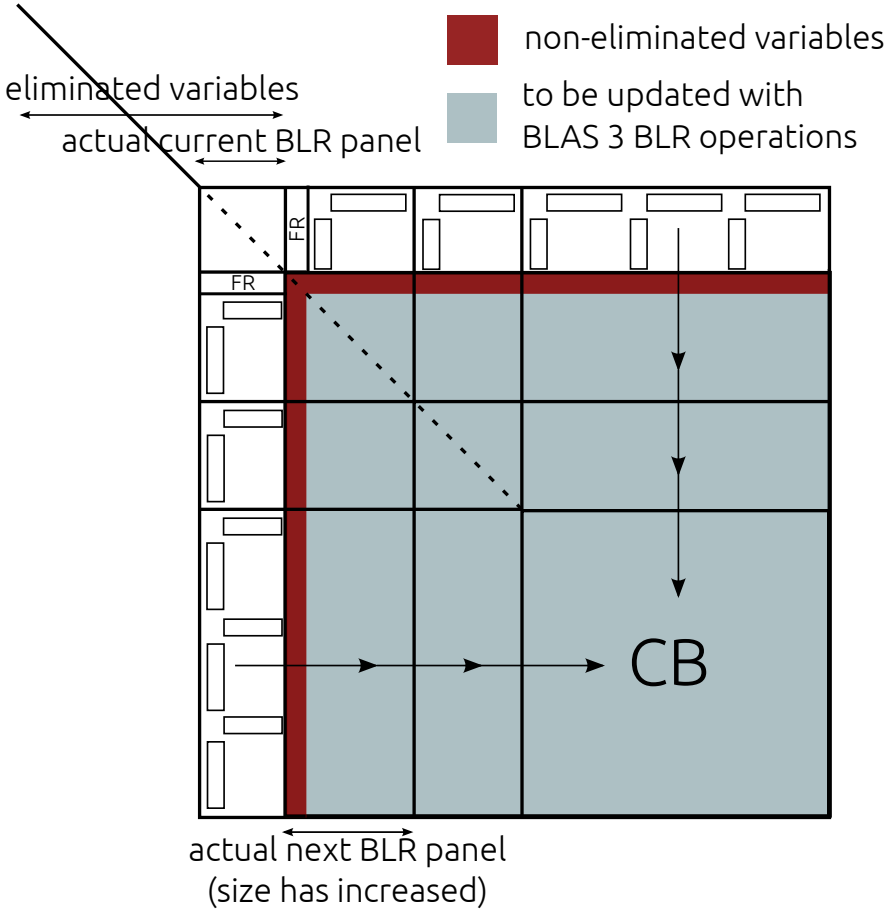


Threshold partial pivoting with BLR



Pivots are delayed panelwise and eventually to the parent node

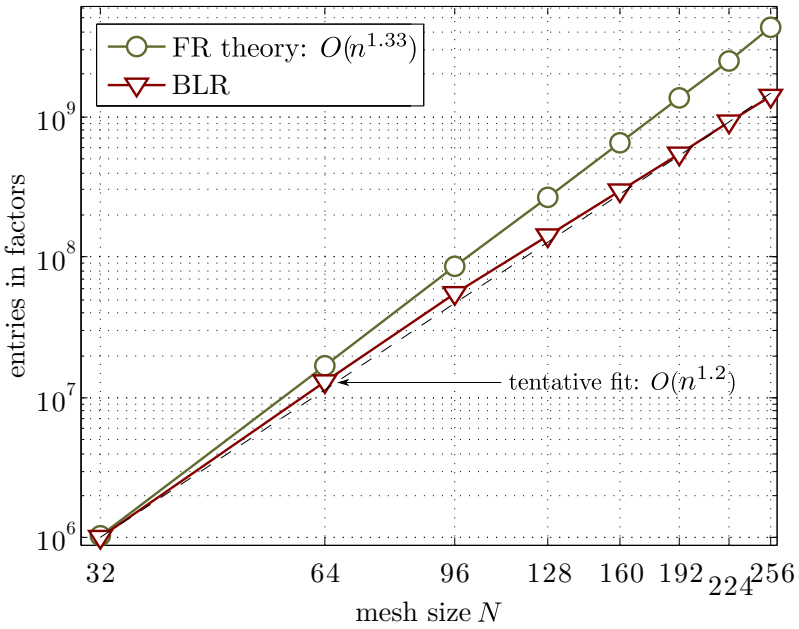
Threshold partial pivoting with BLR



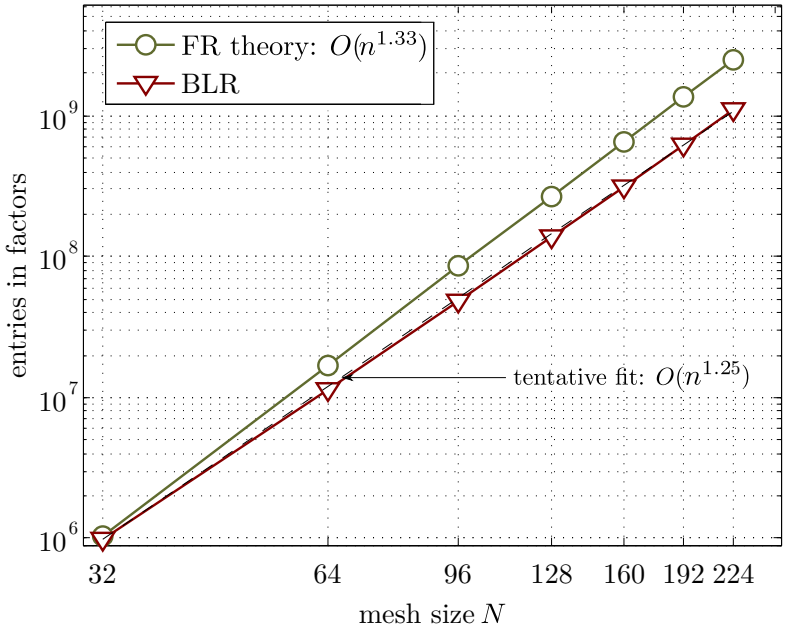
Pivots are delayed panelwise and eventually to the parent node

Experimental MF complexity: entries in factor

Poisson ($\varepsilon = 10^{-14}$) ($n = N^3$) Helmholtz ($\varepsilon = 10^{-8}$)



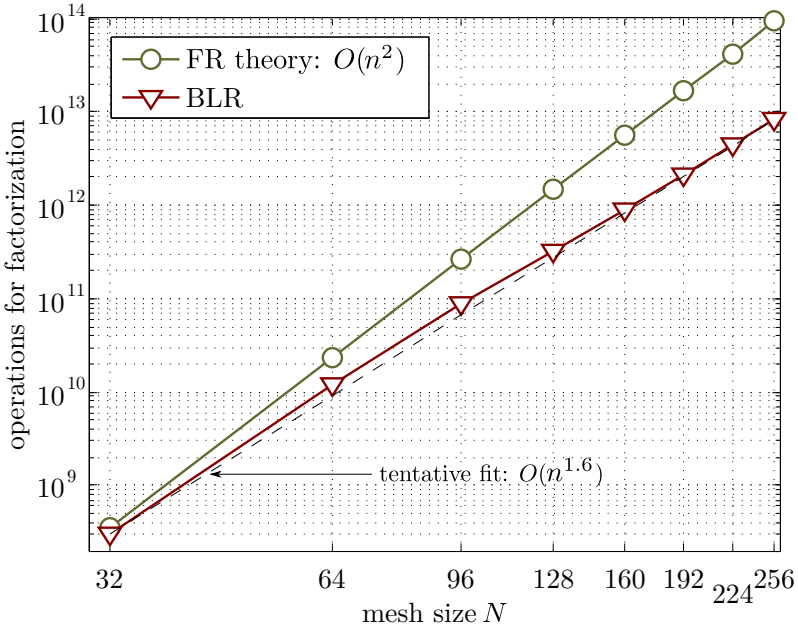
For $N = 256$, 4 times fewer entries in factors



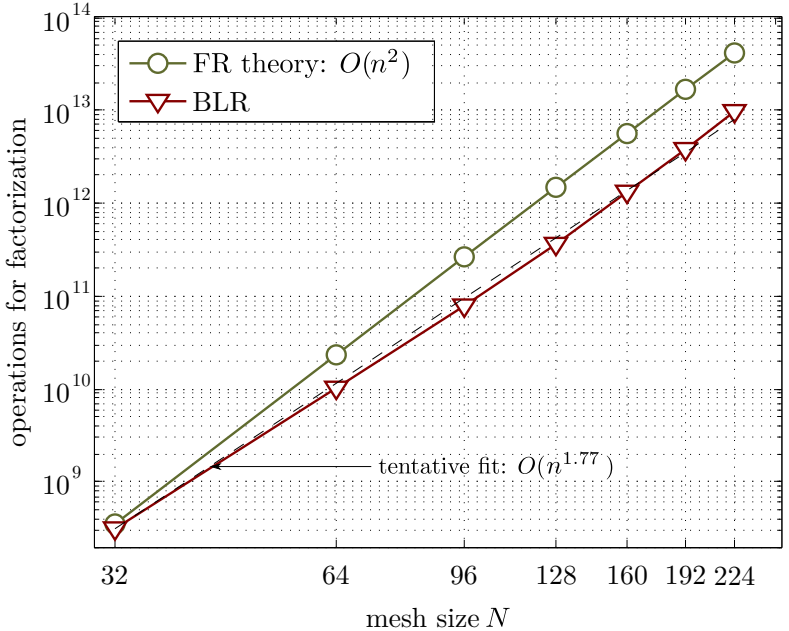
For $N = 256$, 2 times fewer entries in factors

Experimental MF complexity: ops for facto

Poisson ($\varepsilon = 10^{-14}$) ($n = N^3$) Helmholtz ($\varepsilon = 10^{-8}$)



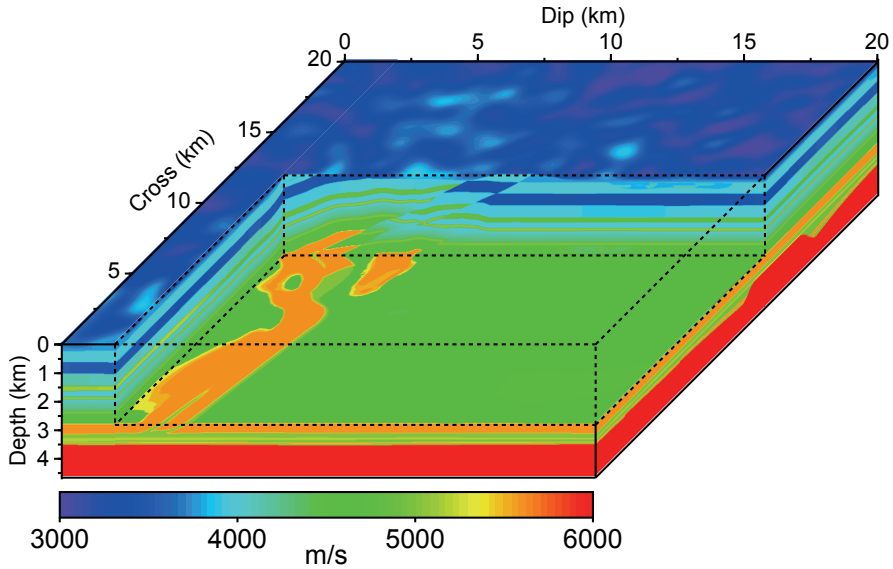
For $N = 256$, 10 times less flops



For $N = 256$, ~ 5 times less flops

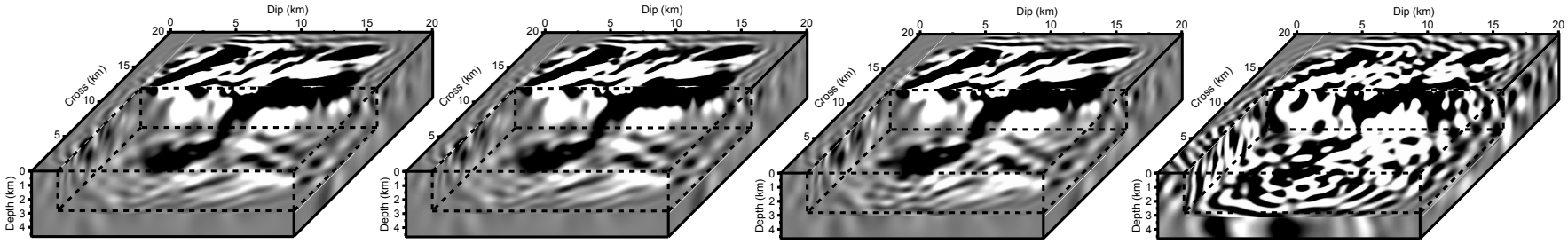
Application to frequency-domain seismic modeling (1)

- Helmholtz equation for seismic modeling (SEISCOPE project)
- EAGE overthrust ground model
- single precision computations



fqcy	Ops LU	Mem LU	Peak memory
2 Hz	8.957E+11	3 GB	4 GB
4 Hz	1.639E+13	22 GB	25 GB
8 Hz	5.769E+14	247 GB	283 GB

Application to frequency-domain seismic modeling (3)



FR 10^{-5} 10^{-4} 10^{-3}

		ops	memory	
ε	fqcy		|L|	|CB|
(10^{-5})	2 Hz	41.8 %	61.8 %	32.3%
	4 Hz	27.4 %	50.0 %	24.4%
	8 Hz	21.8 %	41.6 %	23.9%
(10^{-4})	2 Hz	32.9 %	53.4 %	23.9%
	4 Hz	20.0 %	42.2 %	21.7%
	8 Hz	15.2 %	28.9 %	19.4%

Application to Electromagnetism

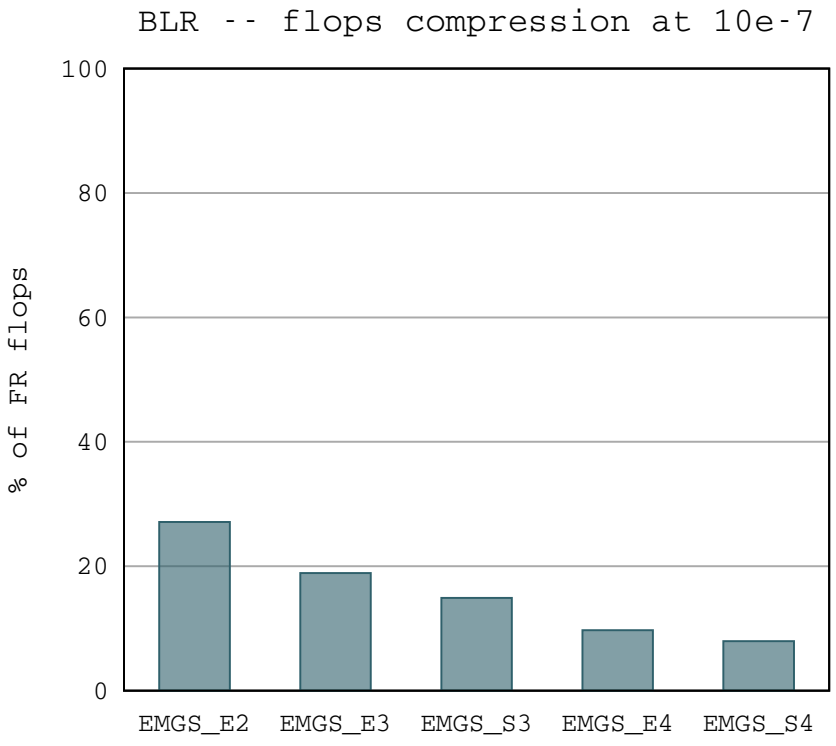
Matrices from EMGS (Norway). All matrices are complex and solved in double-precision

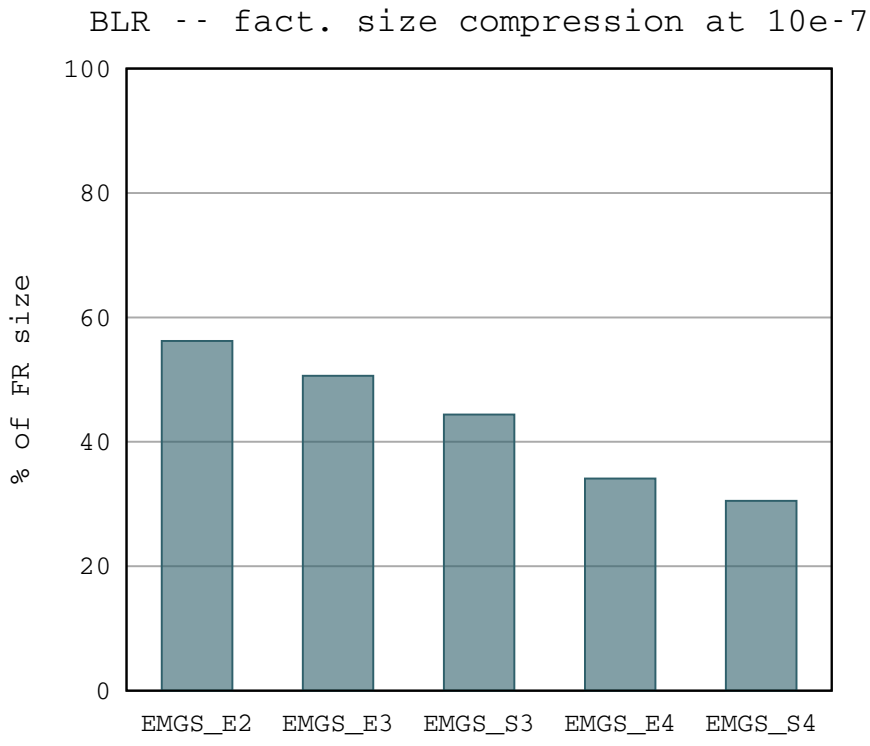
Mat.	n	nnz	factors (GB)	Gflops
EMGS_E2	909312	11658644	16	6.1e+03
EMGS_E3	2885112	37148644	76	5.6e+04
EMGS_S3	3325140	42836538	92	7.5e+04
EMGS_E4	17448276	225626874	897	2.1e+06
EMGS_S4	20590560	266361112	1122	3.0e+06

Experiments are done on the EOS supercomputer at the CALMIP center of Toulouse (grant 2014-P0989):

- Two Intel(r) 10-cores Ivy Bridge 2,8 Ghz and 64 GB memory per node
- Peak per core is 22.4 GFlop/s
- Infiniband interconnect

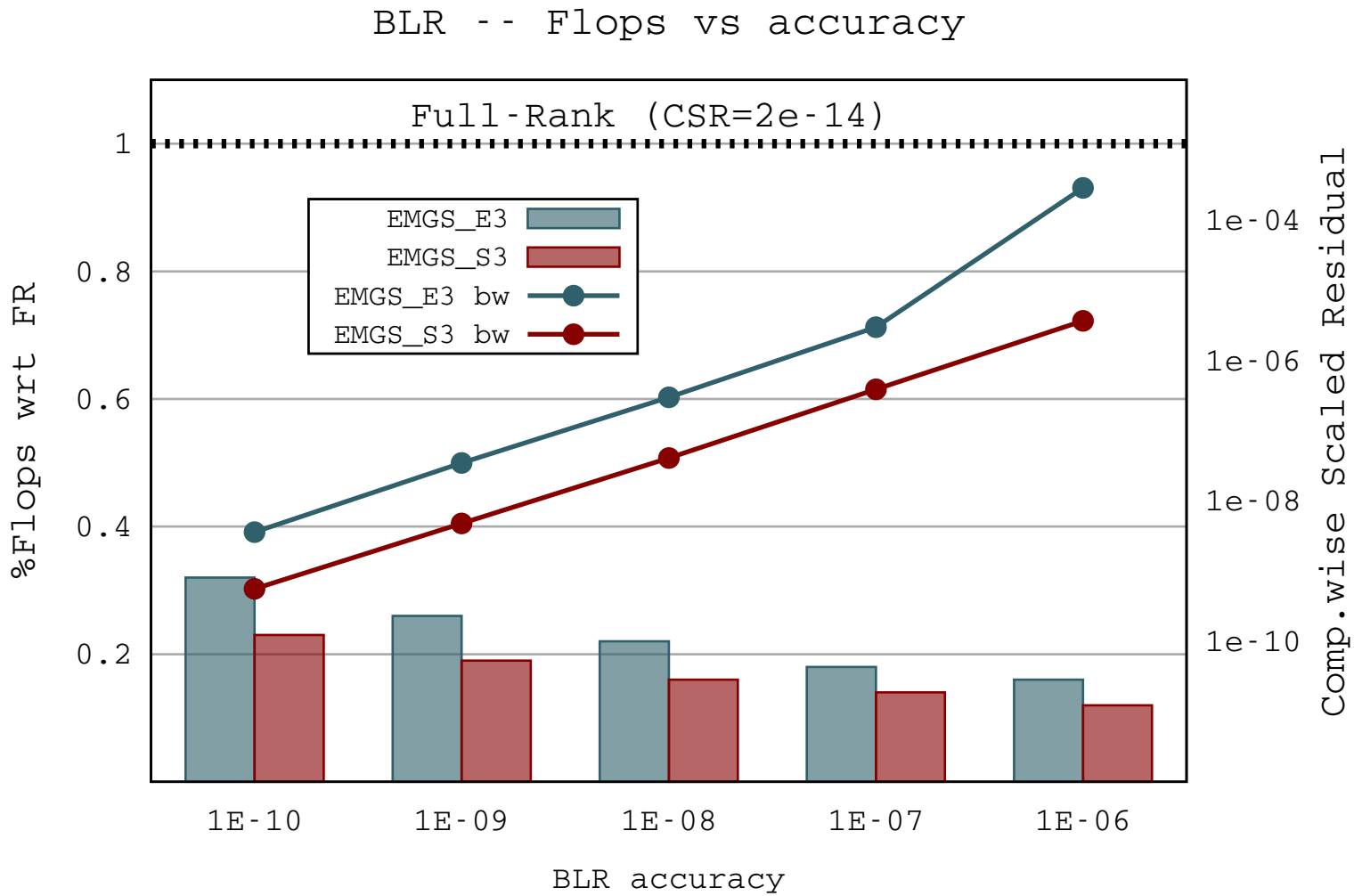
Application to Electromagnetism



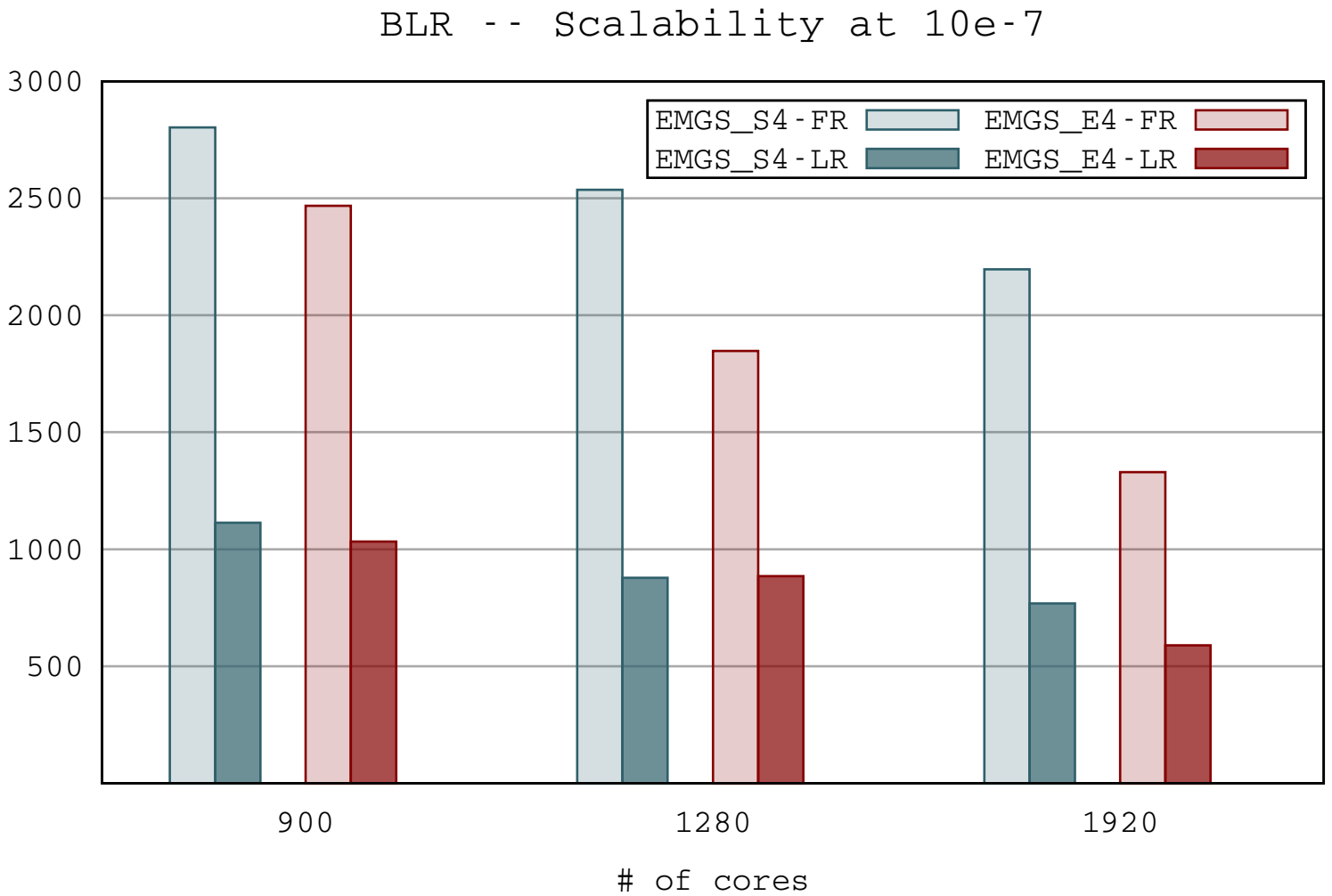


- Gains increase with the size of the problem
- Global memory is reduced more than just factors

Application to Electromagnetism

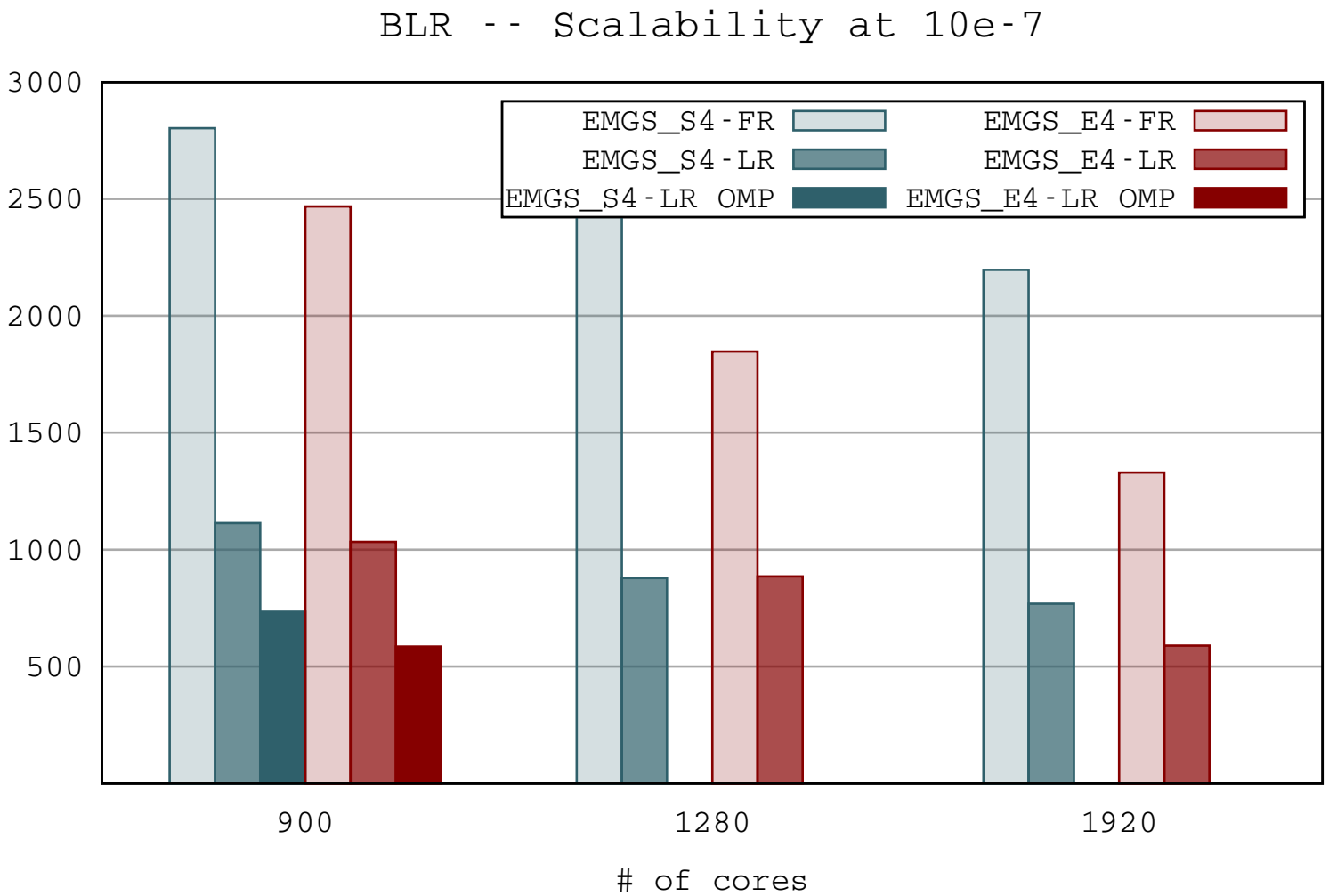


Application to Electromagnetism



- compression overhead
- smaller BLAS granularity (lower seq. and m.threaded speed)
- a factor ~ 2.5 out of ~ 10

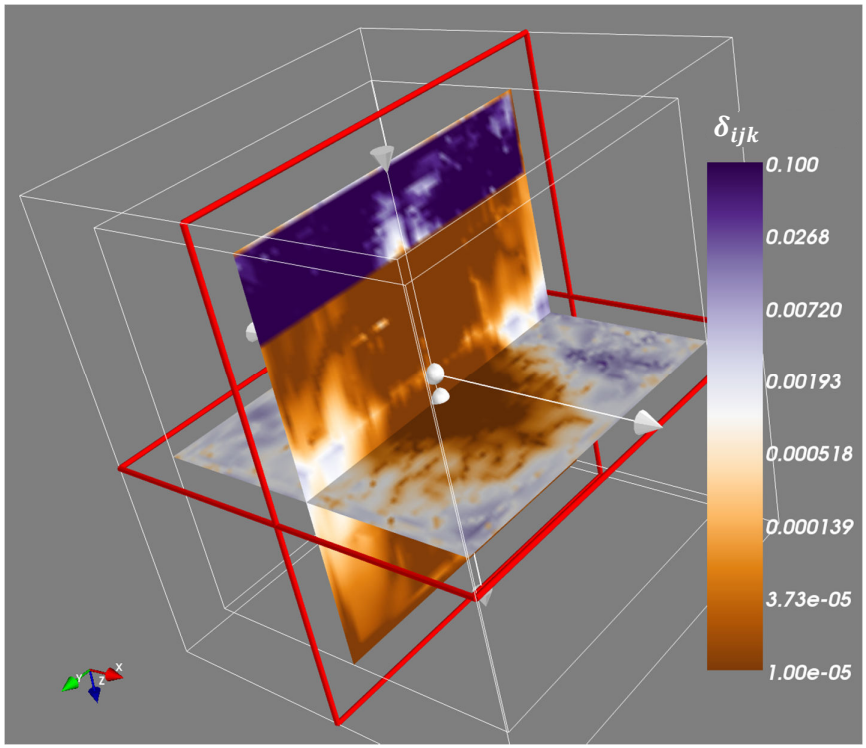
Application to Electromagnetism



- compression overhead
- smaller BLAS granularity (lower seq. and m.threaded speed)
- a factor ~ 4.2 out of ~ 10 thanks to OpenMP

Application to Electromagnetism

E_x, *BLR STRATEGY 2, IR = 0,* $\varepsilon_{BLR} = 10^{-7}$



$$\delta_{ijk} = \sqrt{\frac{|x_{\varepsilon,ijk} - x_{ijk}|^2}{\frac{|x_{\varepsilon,ijk}|^2 + |x_{ijk}|^2}{2} + \eta^2}}$$

(only for E_x)

Relative deviation between Ex-components of low-rank and full-rank solutions

emgs

Thanks!
Questions?