3D frequency-domain seismic modeling with a Parallel BLR multifrontal direct solver

P. Amestoy[1] R. Brossier[2] A. Buttari[1] J.-Y. L'Excellent[1]
T. Mary[1] L. Metivier[2] A. Miniussi[2] S. Operto[2]
J. Virieux[2] C. Weisbecker[3]
[1]MUMPS team [2]SEISCOPE team [3]LSTC

SEG'15, New Orleans Oct. 18-23

Context

SEISCOPE-MUMPS collaboration

- The SEISCOPE consortium investigates high-resolution seismic imaging based on frequency-domain full waveform inversion
- MUMPS is a general purpose parallel sparse direct solver

Two talks

- FWI 6: Renormalization and Direct Nonlinear Inversion
  Stephane Operto's presentation (Room 206, 11:25 AM):
  Efficient 3D frequency-domain full-waveform inversion of ocean-bottom cable data with sparse block low-rank direct solver: A real data case study from the North Sea
- This talk focuses on the linear algebra aspects of the work

Forward problem: a boundary-value (stationary) problem.

$$\left(\frac{\omega^2}{c(x)^2} + \Delta\right) p(x,\omega) = s(x,\omega)$$

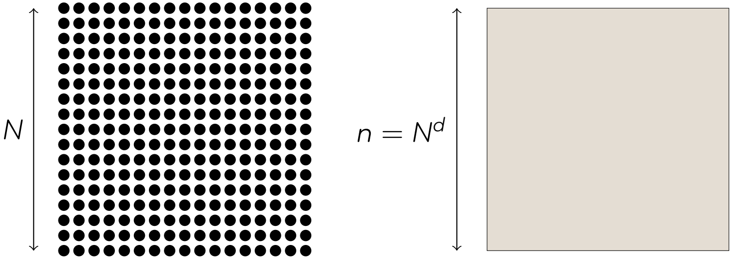
$\Rightarrow$ a large and sparse system of linear equations with multiple right-hand sides.

$$\mathbf{A}(\omega, m, x)\left[\mathbf{p}_1(\omega,x)\mathbf{p}_2(\omega,x)...\mathbf{p}_N(\omega,x)\right] = \left[\mathbf{s}_1(\omega,x)\mathbf{s}_2(\omega,x)...\mathbf{s}_N(\omega,x)\right].$$

Use direct solver to factorize A and solve the system.
Advantages over iterative solvers:

- easy to use (push button $\rightarrow$ get answer)
- numerically robust
- do one factorization and multiple bw/fw substitutions
- can be used to precondition iterative solvers

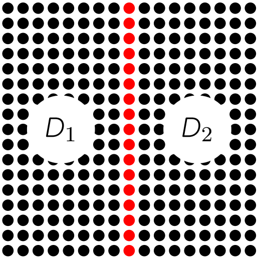
The Multifrontal method

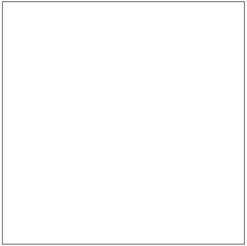


2D problem cost $\propto$

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

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

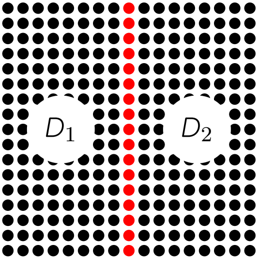


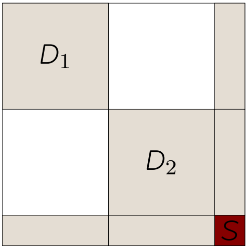


2D problem cost $\propto$

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

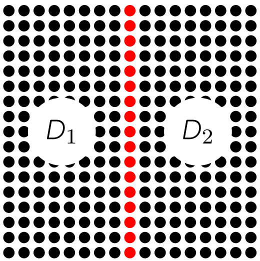
MF (Duff'83) ND (George'73)

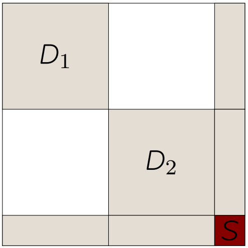




2D problem cost $\propto$

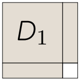
Flops:$\mathcal{O}(N^6)$, mem:$\mathcal{O}(N^4)$



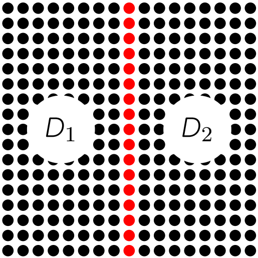


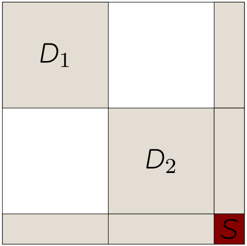
2D problem cost $\propto$

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



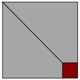
MF (Duff'83) ND (George'73)

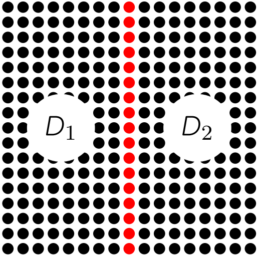


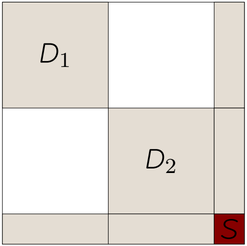


2D problem cost $\propto$

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

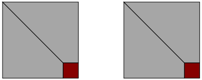


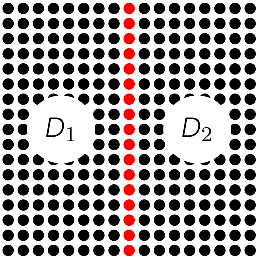


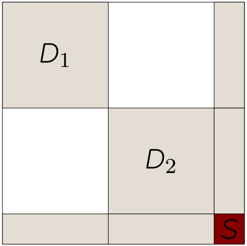


2D problem cost $\propto$

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

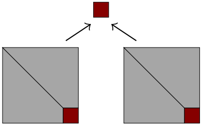


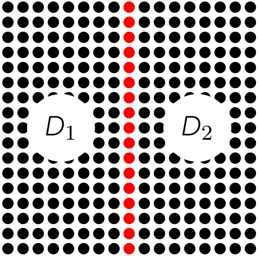


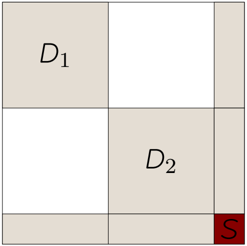


2D problem cost $\propto$

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



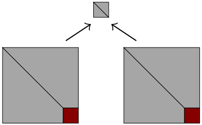


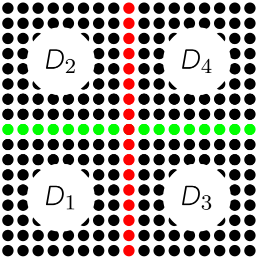


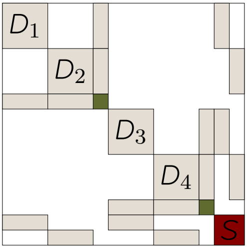
2D problem cost $\propto$

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

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





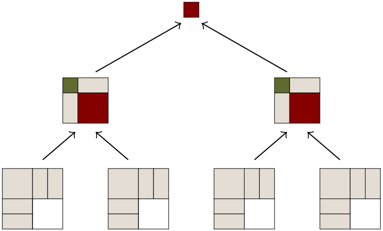


2D problem cost $\propto$

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:$\mathcal{O}(N^2 log(N))$

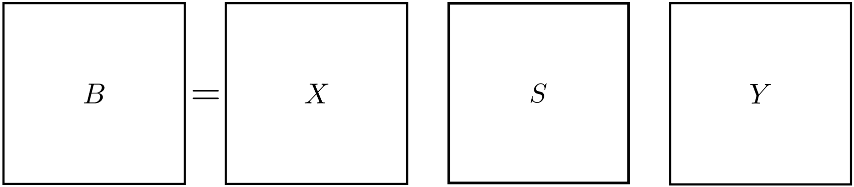


3D problem cost $\propto$

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

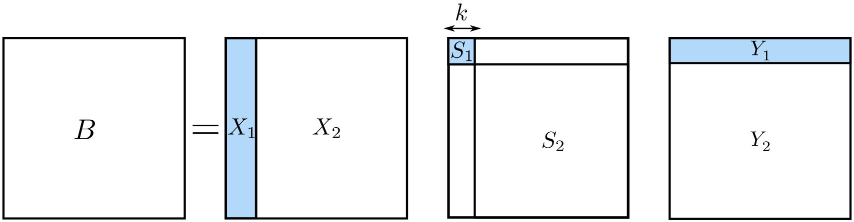
Low-Rank property

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



Low-rank matrices

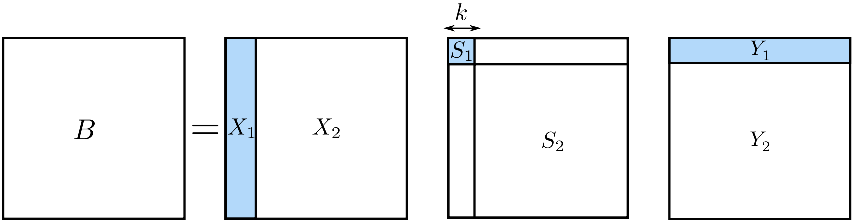
Take a dense matrix B of size $n \times n$ and compute its SVD $B = XSY$:



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

Low-rank matrices

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

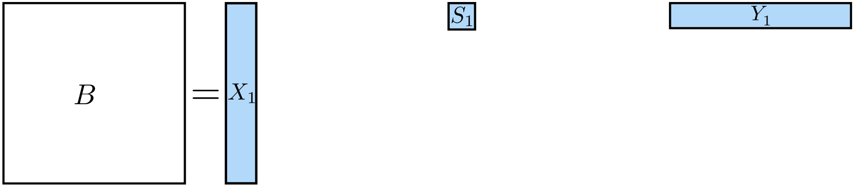


$B = X_1 S_1 Y_1 + X_2 S_2 Y_2 \quad \text{with} \quad S_1(k,k) = \sigma_k > \varepsilon, \; S_2(1,1) = \sigma_{k+1} \leq \varepsilon$

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

Low-rank matrices

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



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

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

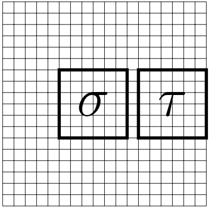
If the singular values of B decay very fast (e.g. exponentially) then $k \ll n$ even for very small ε (e.g. 10^{-14}) $\Rightarrow$ memory and CPU consumption can be reduced considerably with a controlled loss of accuracy ($\leq \varepsilon$) if $\tilde{B}$ is used instead of B

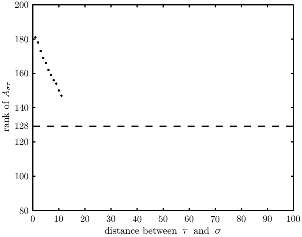
Can we exploit low-rankness in frontal matrices?

Frontal 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 have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low

Can we exploit low-rankness in frontal matrices?

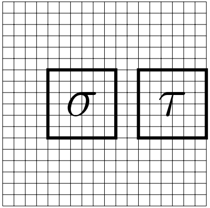
Frontal matrices are usually not low-rank but in many applications they exhibit low-rank blocks.
A block represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low

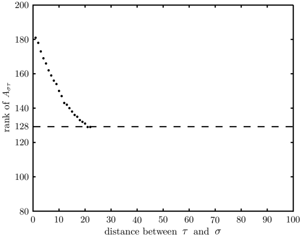




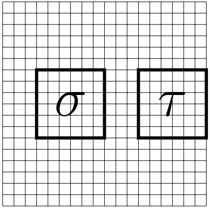
Can we exploit low-rankness in frontal matrices?

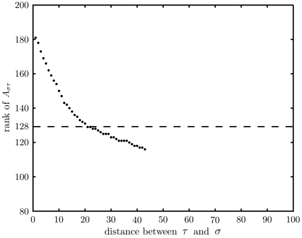
Frontal matrices are usually not low-rank but in many applications they exhibit low-rank blocks.
A block represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low



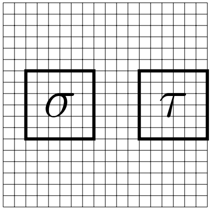


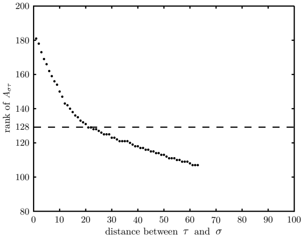
Frontal matrices are usually not low-rank but in many applications they exhibit low-rank blocks.
A block represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low





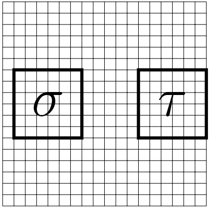
Frontal matrices are usually not low-rank but in many applications they exhibit low-rank blocks.
A block represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low

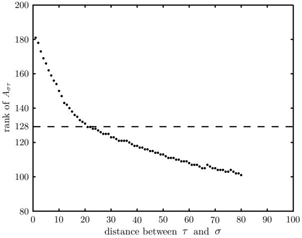




Can we exploit low-rankness in frontal matrices?

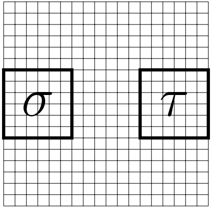
Frontal matrices are usually not low-rank but in many applications they exhibit low-rank blocks.
A block represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low

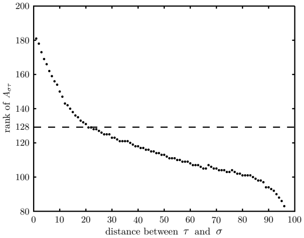




Can we exploit low-rankness in frontal matrices?

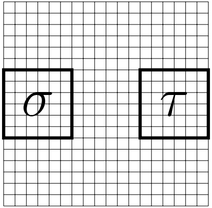
Frontal matrices are usually not low-rank but in many applications they exhibit low-rank blocks.
A block represents the interaction between two subdomains σ and τ. If they have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low

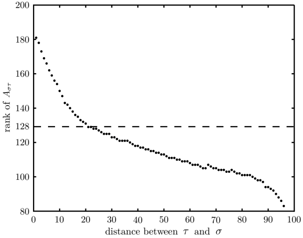




Can we exploit low-rankness in frontal matrices?

Frontal 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 have a small diameter and are far away the interaction is weak $\Rightarrow$ rank is low



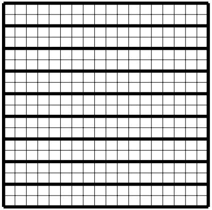


1. compute a clustering of your domain (mesh)
2. permute the matrix accordingly
3. enjoy low-rankness

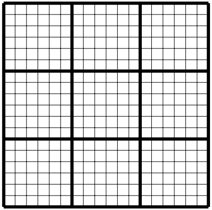
Clustering

Clustering

We aim at a clustering which is such that each frontal matrix has a maximum of low-rank blocks.
If the geometry of the domain, and of the separators is known, the task would be relatively simple



large diameters
small distances



small diameters
large distances

- maximize the relative distance between clusters
- minimize their diameter...
- but not too much to achieve an acceptable BLAS efficiency

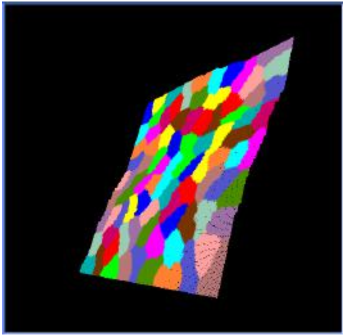
Algebraic clustering/blocking

In a purely algebraic context, we don't have the luxury of knowing the geometry because we only know the matrix
$\rightarrow$ use the adjacency graph instead of the domain geometry

For all the separators
- extract the adjacency graph
- extend it with halo
- pass it to a partitioning tool

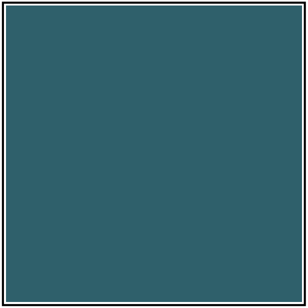
End for

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

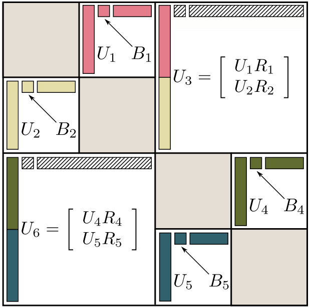


Low-rank formats

Once the blocking is defined, several low-rank formats are possible.

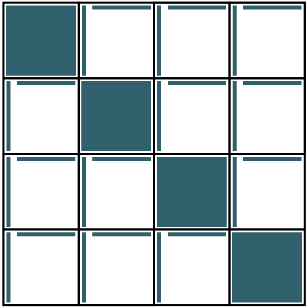


Once the blocking is defined, several low-rank formats are possible.
Some have a hierarchical format ($\mathcal{H}$, $\mathcal{H}^2$, HSS, HODLR, ...)



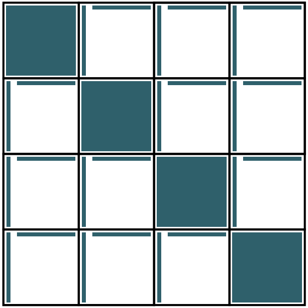
- Leads to very low complexity (fact. is $\sim O(n)$, with a big constant).
- Complex, hierarchical structure.
- Relatively inefficient and expensive SVD/RRQR...(very T&S blocks), unless randomization or low-rank assembly is used.
- Parallelism is difficult to exploit.

Once the blocking is defined, several low-rank formats are possible.
Another one (ours) is Block Low-Rank



- Very simple structure (very little logic to handle).
- Cheap SVD/RRQR.
- Completely parallel.
- Complexity is a question under investigation.

Once the blocking is defined, several low-rank formats are possible.
Another one (ours) is Block Low-Rank



- Very simple structure (very little logic to handle).
- Cheap SVD/RRQR.
- Completely parallel.
- Complexity is a question under investigation.

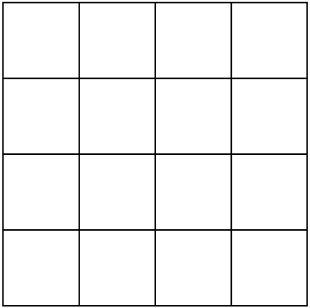
We believe Block Low-Rank (BLR) aims at a good compromise between complexity and performance/usability.

Factorization

BLR LU factorization

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

(b=block size, r=rank)



_GETRF

_TRSM

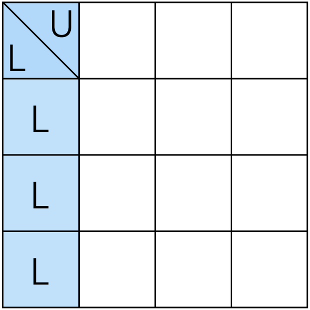
_GEQP3/_GESVD

_GEMM

BLR LU factorization

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

(b=block size, r=rank)

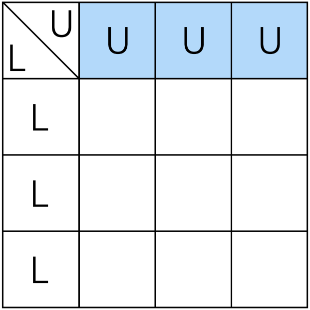


- ▶ _GETRF
- _TRSM
- _GEQP3/_GESVD
- _GEMM

BLR LU factorization

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

(b=block size, r=rank)



_GETRF

▶ _TRSM

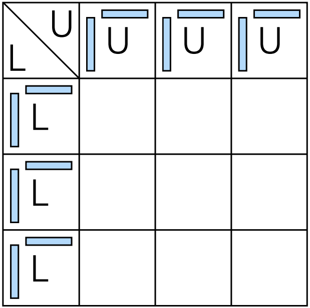
_GEQP3/_GESVD

_GEMM

BLR LU factorization

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

(b=block size, r=rank)



_GETRF

_TRSM

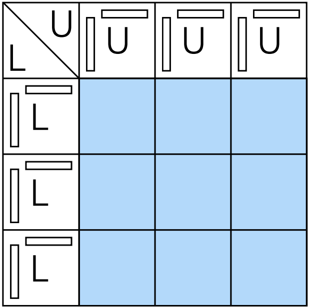
► _GEQP3/_GESVD

_GEMM

BLR LU factorization

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

(b=block size, r=rank)



_GETRF

_TRSM

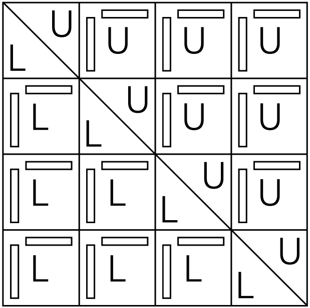
_GEQP3/_GESVD

▶ _GEMM

BLR LU factorization

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

(b=block size, r=rank)



_GETRF

_TRSM

_GEQP3/_GESVD

_GEMM

Experimental results

Setting:

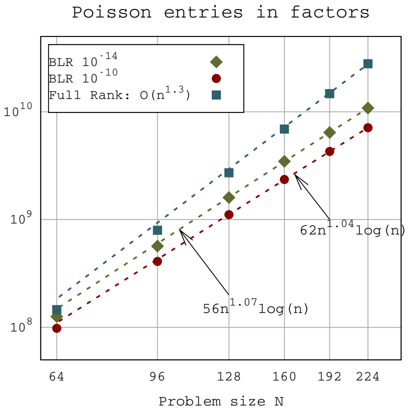
1. **`Poisson`**: N^3 grid with a 7-point stencil with $u = 1$ on the boundary $\partial\Omega$

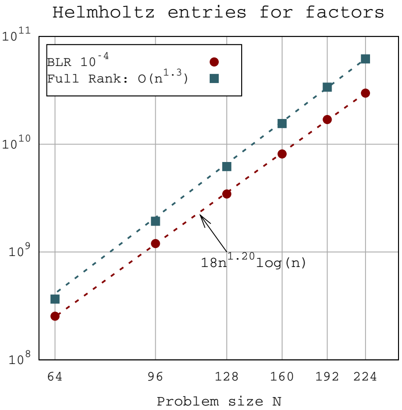
$$\Delta u = f$$

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 $s(x, \omega)$.

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

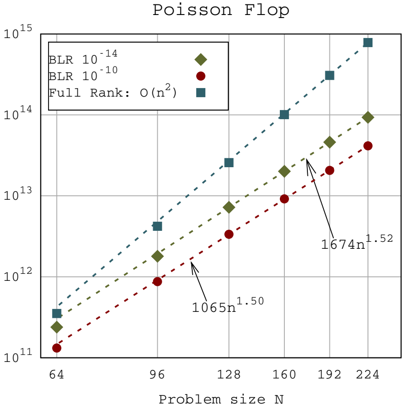
Experimental MF complexity: entries in factor

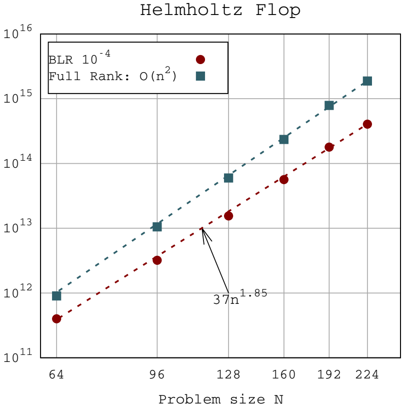




- ε only plays a role in the constant factor
- good agreement with theory
- for Poisson a factor ~ 3 gain with almost no loss of accuracy

Experimental MF complexity: operations





- ε only plays a role in the constant factor
- good agreement with theory
- for Poisson a factor ~ 9 gain with almost no loss of accuracy

Application to frequency-domain seismic modeling

- Credits: SEISCOPE project
- 3D VTI visco-acoustic Valhall model
- VTI visco-acoustic Helmholtz equation

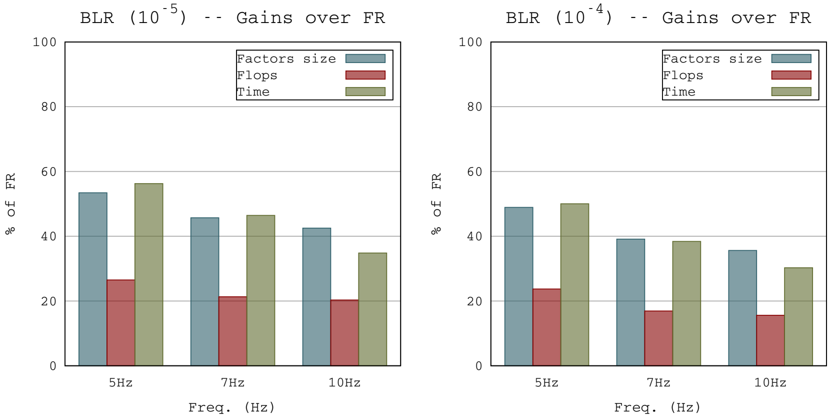
Freq.	n	nnz	factors	flops	time	cores
5Hz	3M	70M	2.5GB	6.5E+13	80s	240
7Hz	7M	177M	6.4GB	4.1E+14	323s	320
10Hz	17M	446M	10.5GB	2.6E+15	1117s	680

Full-rank statistics

Experiments are done on the LICALLO supercomputer at the OCA mesocenter:

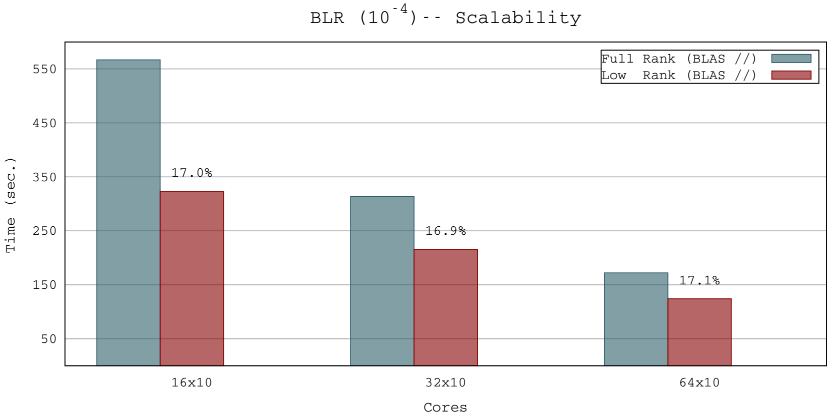
- Two Intel(r) 10-cores Ivy Bridge 2,5 GHz and 64 GB memory
- Peak per core is 20.0 GF/s
- Infiniband FDR interconnect

Application to frequency-domain seismic modeling



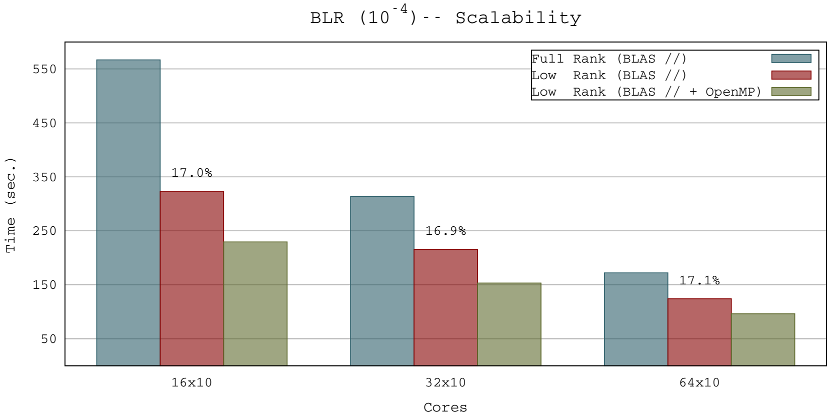
Gains in execution time do not match those in Flops because of the weaker efficiency of BLAS kernels due to the small granularity.

Application to frequency-domain seismic modeling



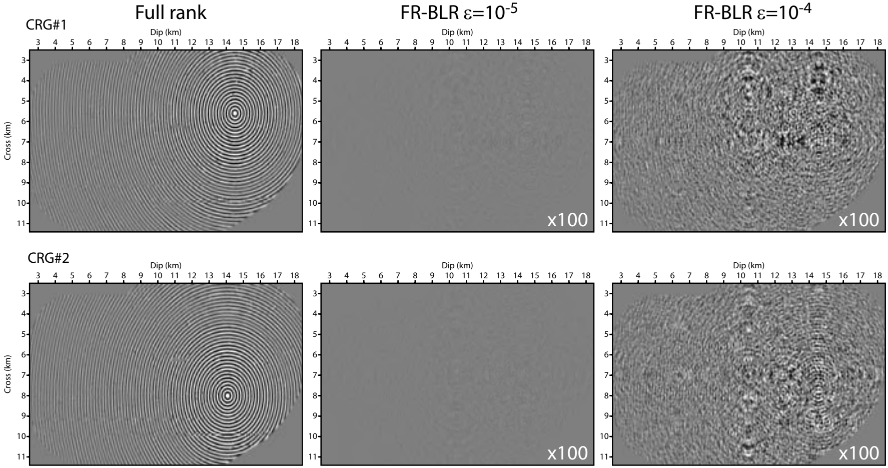
Due to the small size of blocks, multithreaded BLAS is inefficient.

Application to frequency-domain seismic modeling

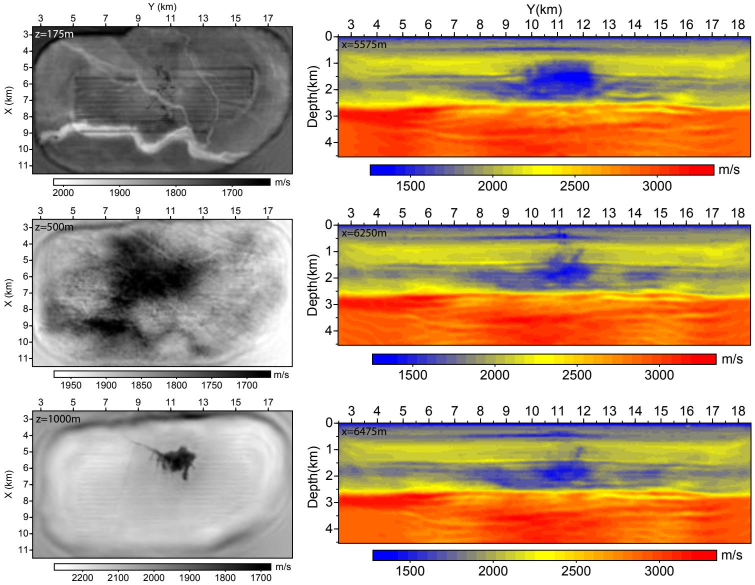


Due to the small size of blocks, multithreaded BLAS is inefficient. We have added OpenMP directives to exploit multicores on BLR computations

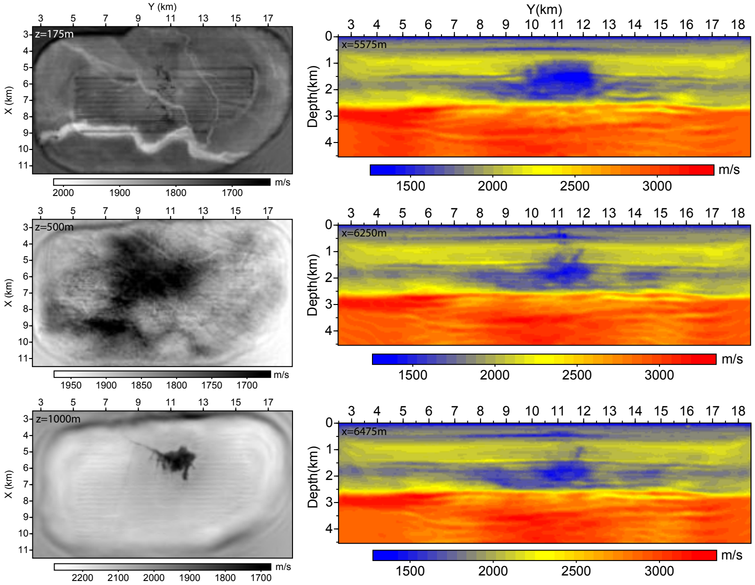
Valhall case study: modeling errors associated with BLR



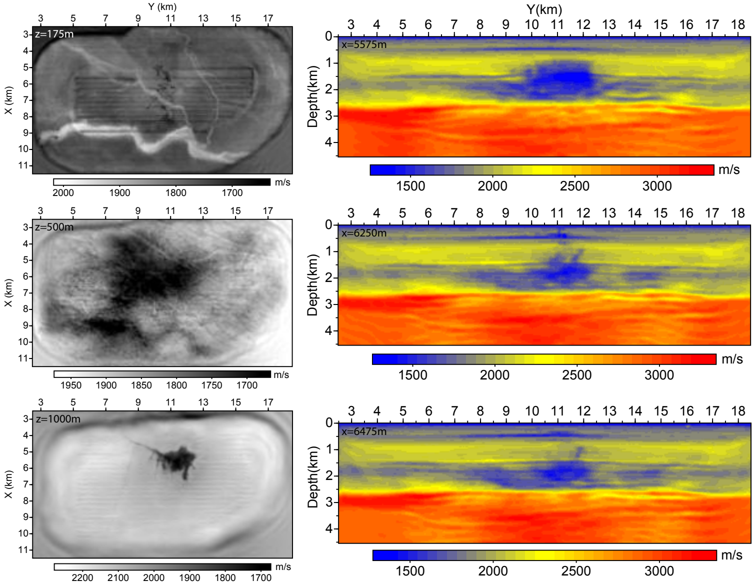
Valhall case study: FWI with FR MUMPS



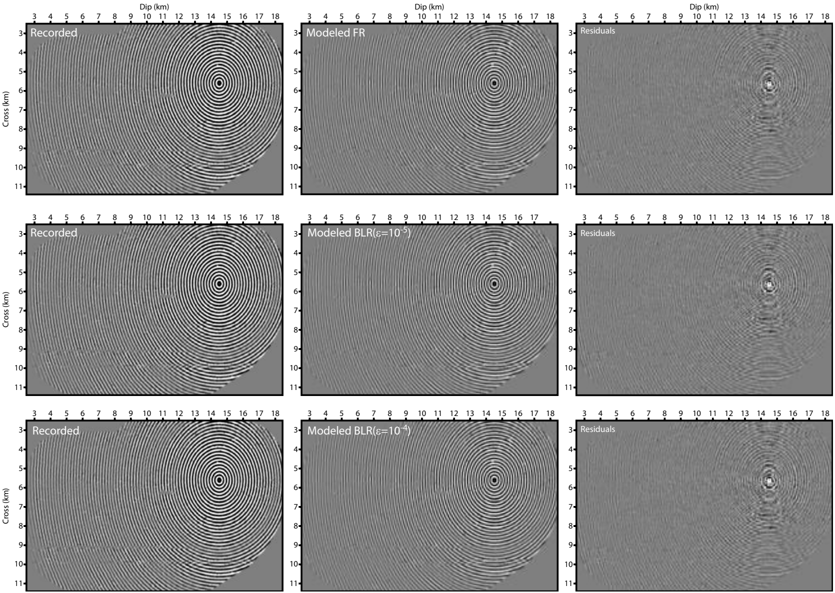
Valhall case study: FWI with MUMPS BLR $\varepsilon = 10^{-5}$



Valhall case study: FWI with MUMPS BLR $\varepsilon = 10^{-4}$



Valhall case study: Data fit - Receiver #1



Solution Phase

Solution phase - more on performance issues

- 1280 Right Hand Sides
- Factorization time: 80s (FR) $\rightarrow$ **47s** (LR)
- Solution time: **193s**

General case $LUX = B$, (X, B centralized and dense)

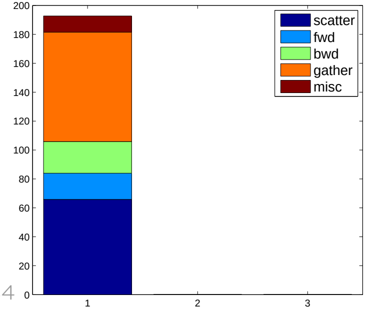
Let NB be the block size
for each block **do**
  Scatter $B_{(1:NB)}$ over all processors
  Compute **Fwd** $Y_{(1:NB)}$: $LY_{(1:NB)} = B_{1:NB}$
  Compute **Bwd** $X_{(1:NB)}$: $UX_{1:NB} = Y_{(1:NB)}$
  Gather $X_{(1:NB)}$ on host processor and postprocess it
end for

Recent improvements of the solution phase

step	
scatter RHS forward backward gather solution	
total	

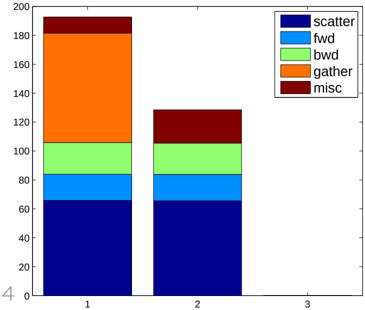
Recent improvements of the solution phase

step	reference
scatter RHS	65.9
forward	18.1
backward	21.9
gather solution	75.6
total	192.7



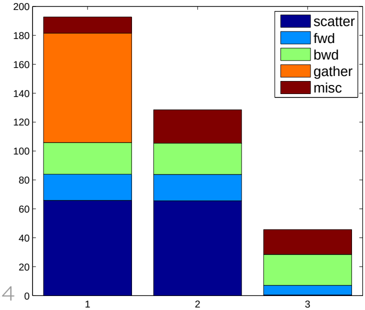
Recent improvements of the solution phase

step	reference	distributed solution
scatter RHS	65.9	65.6
forward	18.1	18.2
backward	21.9	21.6
gather solution	75.6	0.0
total	192.7	128.5



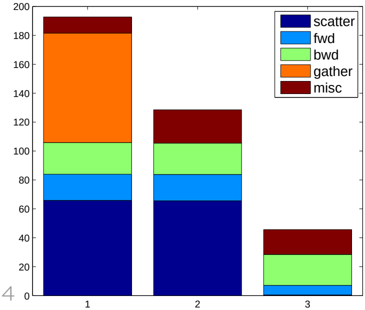
Recent improvements of the solution phase

step	reference	distributed solution	sparse RHS
scatter RHS	65.9	65.6	0.5
forward	18.1	18.2	6.6
backward	21.9	21.6	21.4
gather solution	75.6	0.0	0.0
total	192.7	128.5	45.7



Recent improvements of the solution phase

step	reference	distributed solution	sparse RHS
scatter RHS	65.9	65.6	0.5
forward	18.1	18.2	6.6
backward	21.9	21.6	21.4
gather solution	75.6	0.0	0.0
total	192.7	128.5	45.7



	FR	LR
facto	80s	47s
solve	46s	–

Conclusion and perspectives

Perspectives

- Further improvements of the solution phase:
  - Block-Low-Rank solve
  - Solve-driven scheduling and mapping
  - Multithreading and locality issues with multiple RHS
- Further improvements of the factorization phase:
  - Investigate other variants of BLR LU factorization with better complexity/performance

Acknowledgements

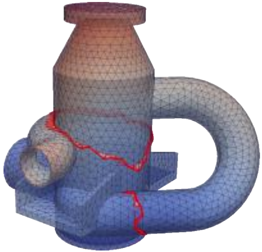
Thanks to the sponsors of

- the SEISCOPE consortium (http://seiscope2.osug.fr)
- and MUMPS consortium (https://mumps-consortium.org)
- This study was granted access to the HPC resources of SIGAMM and CIMENT computer centers and CINES/IDRIS under the allocation 046091 made by GENCI.
- We also thank BP Norge AS and their Valhall partner Hess Norge AS for allowing access to the Valhall data set and initial FWI models.

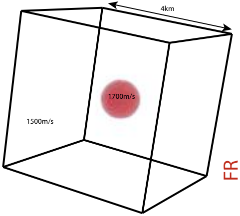
Thanks!
Questions?

Backup Slides

Context



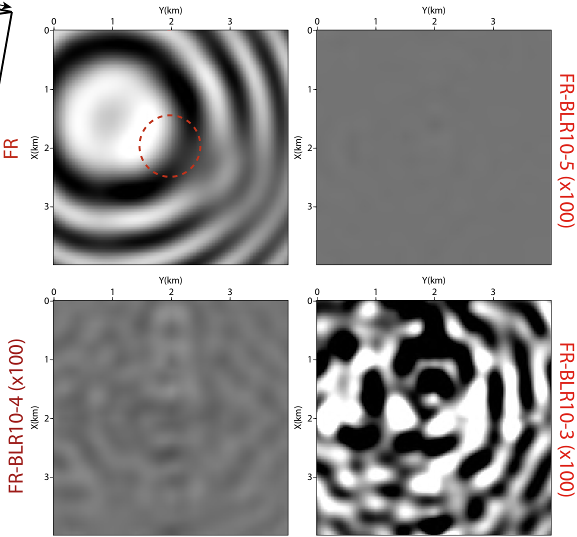
- Started in 2010 following Cleve Ashcraft's presentation at the MUMPS users days
- Initially supported by EDF: one PhD scholarship
- two PhDs: Clement Weisbecker (INPT, EDF, LSTC – 2010-2013), Theo Mary (INPT – 2014-ongoing)
- Several industrial partners/supporters: EDF, EMGS
- Some research collaborators: LBNL, LSTC, SEISCOPE
- Representative publications:
  - C. Weisbecker, P. Amestoy, O. Boiteau, R. Brossier, A. Buttari, J.-Y. L'Excellent, S. Operto and J. Virieux *3D frequency-domain seismic modeling with a Block Low-Rank algebraic multifrontal direct solver*. In: SEG Technical Program Expanded Abstracts, SEG annual meeting, Houston, TX, USA. DOI: 10.1190/segam2013-0603.1. 2013
  - P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, and C. Weisbecker *Improving multifrontal methods by means of block low-rank representations*. To appear on SIAM J. Scientific Computing



Anisotropic model
Vp0=1.5km/s/1.7km/s
δ=0.05,ε=0.1

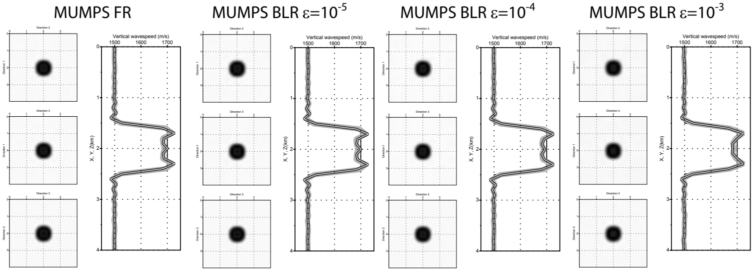
Transmission acquisition
7x7 shots on each face
41x41 receivers on the opposite face

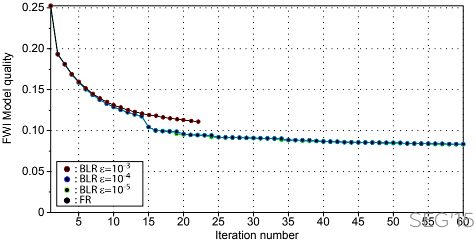
Single frequency modeling/inversion (4Hz)



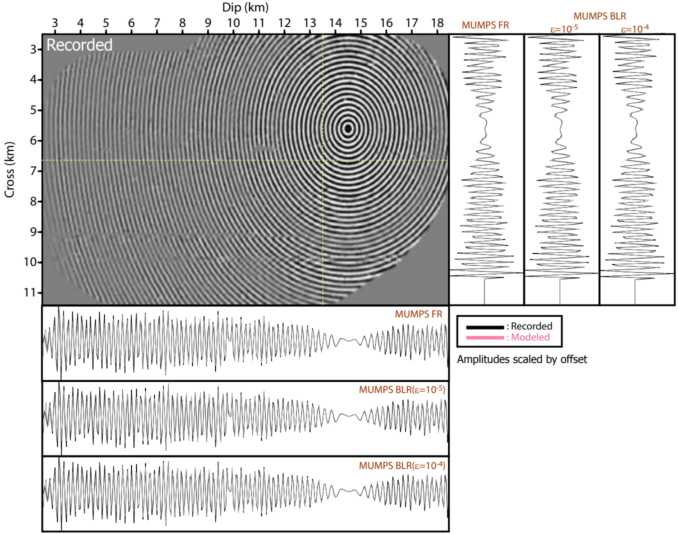
Inclusion model: FWI with BLR MUMPS

- Single frequency inversion (4Hz). Transmission experiment (7 x 7 shots on each face; 41 x 41 receivers on the opposite face).
- Note line-search failure at iteration 22 for $\varepsilon = 10^{-3}$.





Valhall case study: Data fit - Receiver #1



Complexity of BLR LU factorization

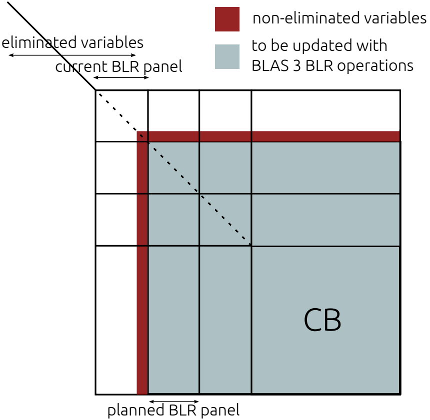
Depending on when and how the compression is done, different variants are possible with different theoretical complexity:

	operations		memory	
	$r = O(1)$	$r = O(N)$	$r = O(1)$	$r = O(N)$
FR	$O(n^2)$	$O(n^2)$	$O(n^{\frac{4}{3}})$	$O(n^{\frac{4}{3}})$
BLR FSCU	$O(n^{\frac{5}{3}})$	$O(n^{\frac{11}{6}})$	$O(n \log n)$	$O(n^{\frac{4}{3}})$
BLR FCSU	$O(n^{\frac{14}{9}})$	$O(n^{\frac{16}{9}})$	$O(n \log n)$	$O(n^{\frac{4}{3}})$
BLR FSCU+LUA	$O(n^{\frac{14}{9}})$	$O(n^{\frac{16}{9}})$	$O(n \log n)$	$O(n^{\frac{4}{3}})$
BLR FCSU+LUA	$O(n^{\frac{4}{3}})$	$O(n^{\frac{5}{3}} \log n)$	$O(n \log n)$	$O(n^{\frac{4}{3}})$
$\mathcal{H}$	$O(n^{\frac{4}{3}})$	$O(n^{\frac{5}{3}})$	$O(n)$	$O(n^{\frac{7}{6}})$
$\mathcal{H}$ (fully struct.)	$O(n)$	$O(n^{\frac{4}{3}})$	$O(n)$	$O(n^{\frac{7}{6}})$

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

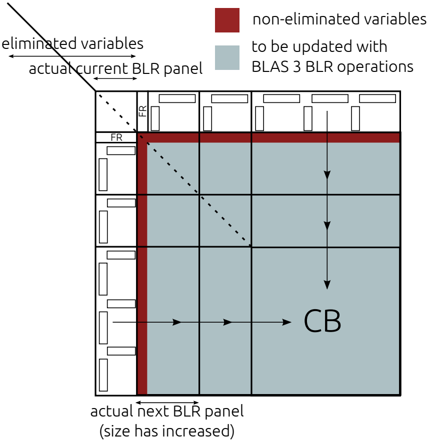
If updates are accumulated and applied at once (LUA), a further reduction can be achieved which leads to the same theoretical complexity as $\mathcal{H}$.

Threshold partial pivoting with BLR



Pivots are delayed panelwise and eventually to the parent node

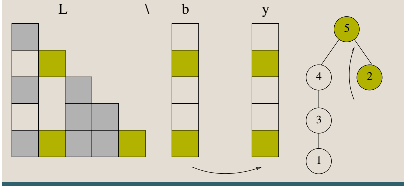
Threshold partial pivoting with BLR



Pivots are delayed panelwise and eventually to the parent node

Exploiting sparsity to reduce flops during solve

solve $y \leftarrow L \setminus b$



- In case of sparse RHS only part of factors/operations needs to be loaded/performed
- Objectives with sparse RHS
  - Efficient use of the RHS sparsity
  - Characterize L and U factors to be loaded
  - Characterize operations to be performed

1. Predicting structure of the solution vector, *Gilbert-Liu*, '93
2. Note that solving with sparse RHS on irreducible matrices can only impact the performance of the forward phase : $Ly = b$.