Grouping variables in Frontal Matrices to improve Low-Rank Approximations in a Multifrontal Solver

Clément Weisbecker
ENSEEIHT-IRIT, Université de Toulouse, France

Preconditioning 2011, May 16, 2011

Joint work with Patrick Amestoy , Cleve Ashcraft, Olivier Boiteau, Alfredo Buttari and Jean-Yves L'Excellent.

PhD started on October 1st, 2010 and financed by EDF.

Grouping variables in Frontal Matrices to improve Low-Rank Approximations in a Multifrontal Solver

Grouping variables in Frontal Matrices to improve Low-Rank Approximations in a Multifrontal Solver

- Multifrontal solver :
  - direct solver for large linear systems
  - well known and studied

Grouping variables in Frontal Matrices to improve Low-Rank Approximations in a Multifrontal Solver

- Multifrontal solver :
  - direct solver for large linear systems
  - well known and studied
- Low-rank approximations :
  - already used in several areas for data compression
  - accuracy controlled by a numerical parameter
  - interesting algebraic features

Grouping variables in Frontal Matrices to improve Low-Rank Approximations in a Multifrontal Solver

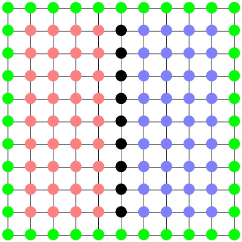
- Multifrontal solver :
  - direct solver for large linear systems
  - well known and studied
- Low-rank approximations :
  - already used in several areas for data compression
  - accuracy controlled by a numerical parameter
  - interesting algebraic features

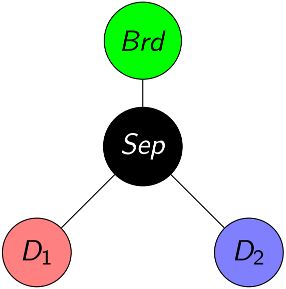
⇒ Try to combine these two notions to improve multifrontal solvers, in particular the `MUMPS` multifrontal solver

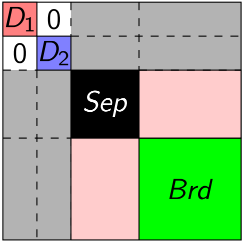
The multifrontal method

I. S. Duff and J. K. Reid, The multifrontal solution of indefinite sparse symmetric linear systems, ACM Transactions on Mathematical Software, 1983.

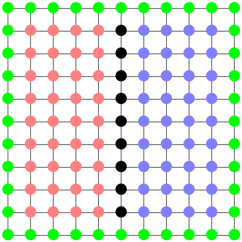
Idea

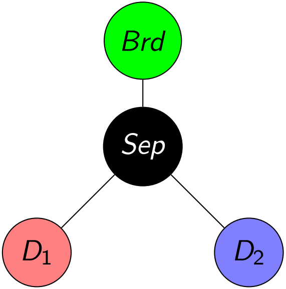


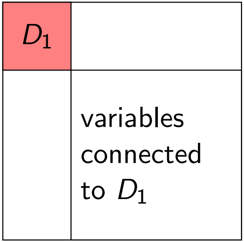


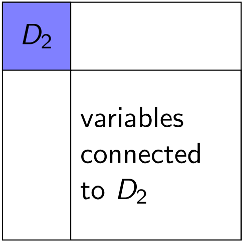


Idea

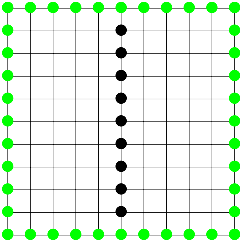


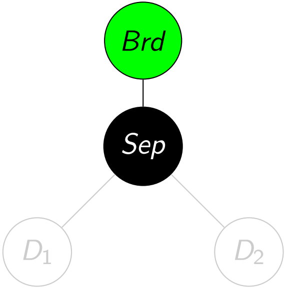

Brd
Sep
D_1
D_2


D_1
variables connected to D_1

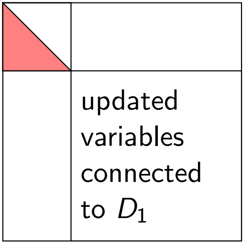

D_2
variables connected to D_2

Idea

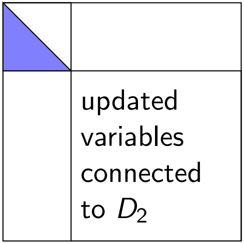




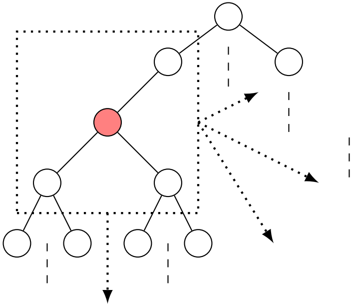
Brd
Sep
D_1
D_2



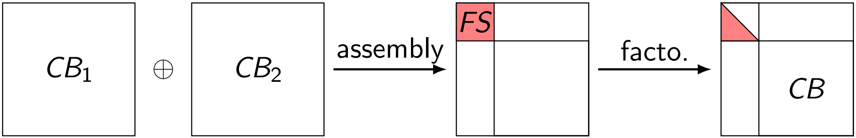
updated variables connected to D_1



updated variables connected to D_2



At each node, an incomplete factorization of the frontal matrix is performed :



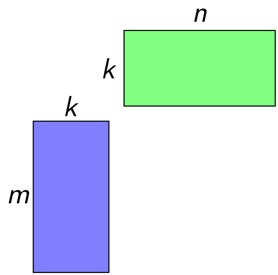
Low-rank theory

General idea

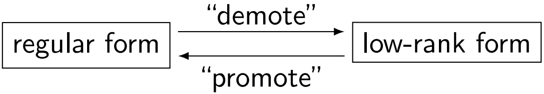
Outer-product form

Let $A \in \mathbb{R}^{m \times n}$ be a matrix of rank k . Let $U \in \mathbb{R}^{m \times k}$ and $V \in \mathbb{R}^{n \times k}$ be two matrices. The *outer-product form* of A is :

$$A = UV^T$$



- storage : $k(m+n)$ vs mn
- if $k < \frac{mn}{m+n} \longrightarrow$ low-rank form



Advantages and Drawbacks

- **Drawbacks**
  - Each outer-product form requires the computation of a *SVD* or a *QR* decomposition
  - More sophisticated data to store and manipulate (Householder vectors)

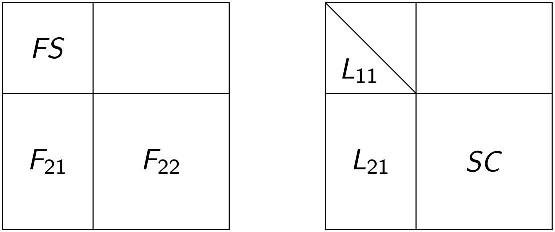
- **Advantages**
  - Reduction of the quantity of information stored
  - Basic algebra operations can be done more efficiently
  - Accuracy of the approximation directly controlled by a numerical parameter

Implementation of low-rank methods within a multifrontal solver

- J. Xia, S. Chandrasekaran, M. Gu and X. S. Li, Superfast Multifrontal Method for Large Structured Linear Systems of Equations, SIAM Journal on Matrix Analysis and Applications.
- M. Bebendorf, Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems, Springer.
- L. Grasedyck, R. Kriemann and S. Le Borne, Parallel black box $\mathcal{H}$-LU preconditioning for elliptic boundary value problems, Computing and Visualization in Science.

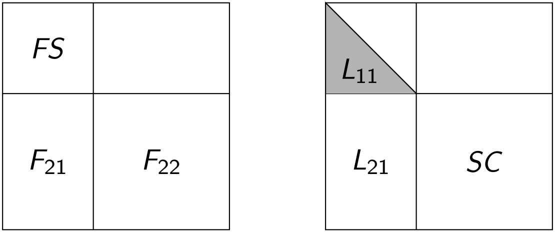
Complete front processing (Cholesky)

- " FSU " factorization of the front :



Complete front processing (Cholesky)

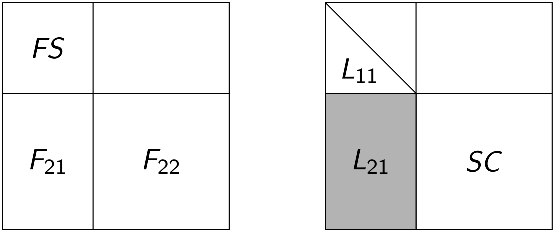
- " FSU " factorization of the front :



1. **F**actor: $FS = L_{11} \cdot L_{11}^{T}$ (Cholesky factorization)

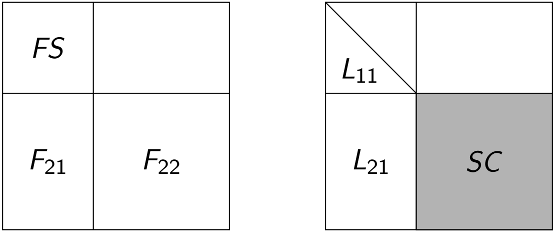
Complete front processing (Cholesky)

- " FSU " factorization of the front :



1. **F**actor: $FS = L_{11} \cdot L_{11}^T$ (Cholesky factorization)
2. **S**olve (TRSM): $L_{21} = F_{21} \cdot L_{11}^{-T}$

- " FSU " factorization of the front :



1. **F**actor: $FS = L_{11} \cdot L_{11}^T$ (Cholesky factorization)
2. **S**olve (TRSM): $L_{21} = F_{21} \cdot L_{11}^{-T}$
3. **U**pdate (SYRK): $SC = F_{22} - L_{21} \cdot L_{21}^T$

Link with the low-rank approximations

The UPDATE(SYRK) and SOLVE(TRSM) phases can be performed using low-rank operations !

- LR-**U**pdate $(L_{21} = U \cdot V^T)$: $SC = F_{22} - U \cdot (V^T \cdot V) \cdot U^T$
- LR-**S**olve $(F_{21} = U \cdot V^T)$: $L_{21} = U \cdot (V^T \cdot L_{11}^{-T})$

Link with the low-rank approximations

The UPDATE(SYRK) and SOLVE(TRSM) phases can be performed using low-rank operations !

- LR-**U**pdate $(L_{21} = U \cdot V^T)$: $SC = F_{22} - U \cdot (V^T \cdot V) \cdot U^T$
- LR-**S**olve $(F_{21} = U \cdot V^T)$: $L_{21} = U \cdot (V^T \cdot L_{11}^{-T})$

Problem :

The fronts of the multifrontal tree are FULL rank

Link with the low-rank approximations

The UPDATE(SYRK) and SOLVE(TRSM) phases can be performed using low-rank operations !

- LR-**U**pdate $(L_{21} = U \cdot V^T)$: $SC = F_{22} - U \cdot (V^T \cdot V) \cdot U^T$
- LR-**S**olve $(F_{21} = U \cdot V^T)$: $L_{21} = U \cdot (V^T \cdot L_{11}^{-T})$

Problem :

The fronts of the multifrontal tree are FULL rank

S. Chandrasekaran, P. Dewilde, M. Gu and N. Somasunderam, On the Numerical Rank of the Off- Diagonal Blocks of Schur Complements of Discretized Elliptic PDEs, SIAM Journal on Matrix Analysis and Applications.

Link with the low-rank approximations

The UPDATE(SYRK) and SOLVE(TRSM) phases can be performed using low-rank operations !

- LR-**U**pdate $(L_{21} = U \cdot V^T)$: $SC = F_{22} - U \cdot (V^T \cdot V) \cdot U^T$
- LR-**S**olve $(F_{21} = U \cdot V^T)$: $L_{21} = U \cdot (V^T \cdot L_{11}^{-T})$

Problem :

The fronts of the multifrontal tree are FULL rank

S. Chandrasekaran, P. Dewilde, M. Gu and N. Somasunderam, On the Numerical Rank of the Off- Diagonal Blocks of Schur Complements of Discretized Elliptic PDEs, SIAM Journal on Matrix Analysis and Applications.

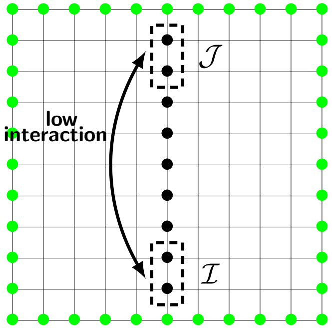
Solution :

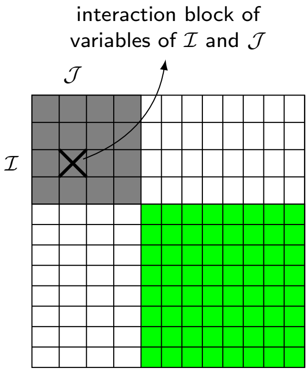
Group variables to obtain low-rank *subblocks*

The low-rank method

"How to find low-rank subblocks ?"

$$\boxed{min\{diam(\mathcal{I}), diam(\mathcal{J})\} < \eta \cdot dist(\mathcal{I}, \mathcal{J})}$$ *(Bebendorf)*



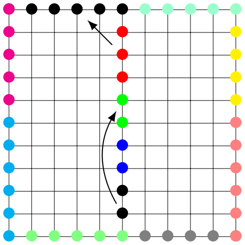


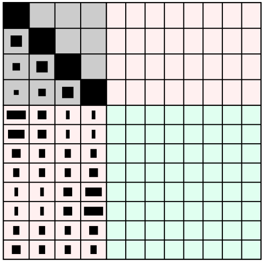
Variables of $\mathcal{I}$ and $\mathcal{J}$ well separated $\Rightarrow$ $L(\mathcal{I}, \mathcal{J})$ has a low numerical rank

Objective : define groups of well separated variables

First way : geometric partitioning

- geometric reordering : Geometric properties are taken into account
- Laplacian problem on square 500 × 500 domain



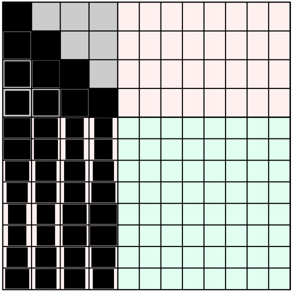


(density proportional to the rank)

Objective : define groups of well separated variables

Second way : random partitioning

- Random reordering : geometry is not taken into account
- Laplacian problem on square 500 × 500 domain



(density proportional to the rank)

Grouping algorithm

$\Rightarrow$ VERY important to have a good grouping of the variables

Grouping algorithm

⇒ VERY important to have a good grouping of the variables

Implemented algorithm :

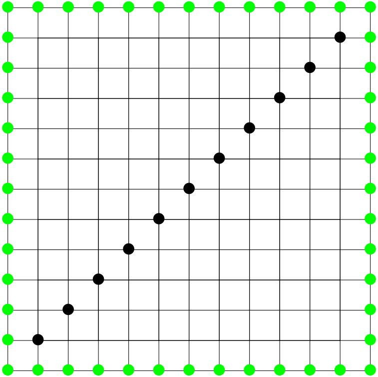
- Halo-based algorithm to catch the geometry
- Coupled with a third party partitioning tool

Grouping algorithm

⇒ VERY important to have a good grouping of the variables

Implemented algorithm :

- Halo-based algorithm to catch the geometry
- Coupled with a third party partitioning tool



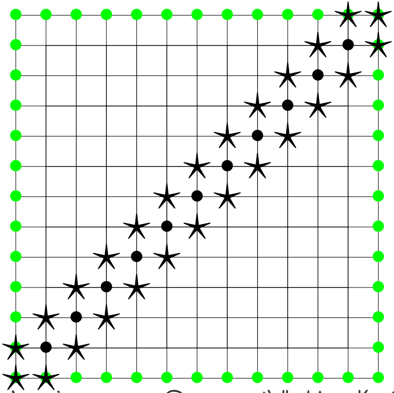
1. The separator

Grouping algorithm

⇒ VERY important to have a good grouping of the variables

Implemented algorithm :

- Halo-based algorithm to catch the geometry
- Coupled with a third party partitioning tool



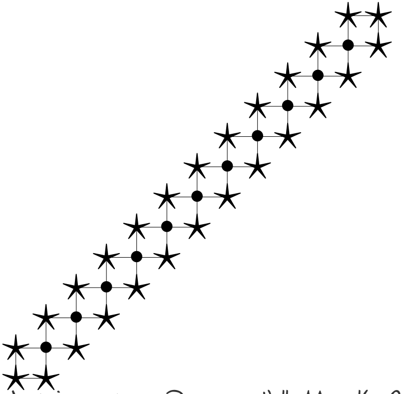
1. The separator
2. The halo

Grouping algorithm

⇒ VERY important to have a good grouping of the variables

Implemented algorithm :

- Halo-based algorithm to catch the geometry
- Coupled with a third party partitioning tool

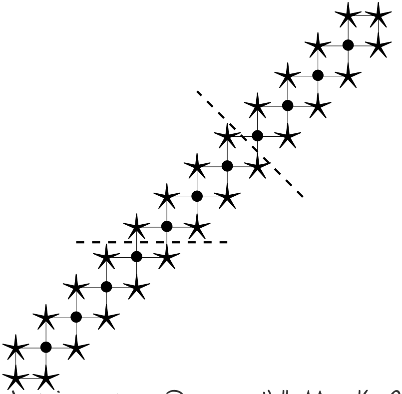


1. The separator
2. The halo
3. Extraction of the halo

⇒ VERY important to have a good grouping of the variables

Implemented algorithm :

- Halo-based algorithm to catch the geometry
- Coupled with a third party partitioning tool

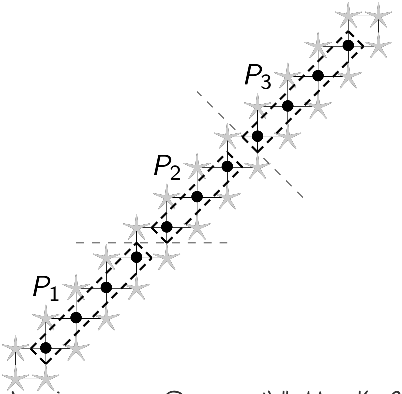


1. The separator
2. The halo
3. Extraction of the halo
4. Partition of the halo

⇒ VERY important to have a good grouping of the variables

Implemented algorithm :

- Halo-based algorithm to catch the geometry
- Coupled with a third party partitioning tool



1. The separator
2. The halo
3. Extraction of the halo
4. Partition of the halo
5. Partition of the separator

Strategies

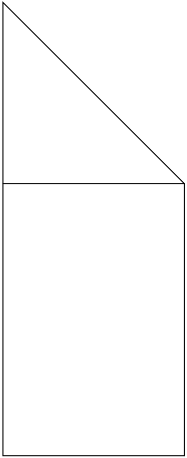
Remains to define how to use the low-rank operations within a front

$\Rightarrow$ 4 strategies to process a front :

- Strategy FSU**D**
- Strategy FS**D**U
- Strategy **panel** FS**D**U
- (Strategy F**D**SU)

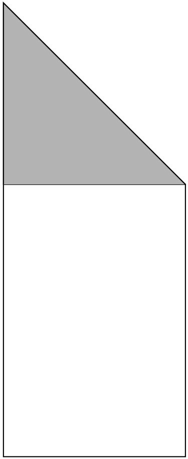
Note : we do not use low-rank within the Schur complements yet

Strategy FSUD



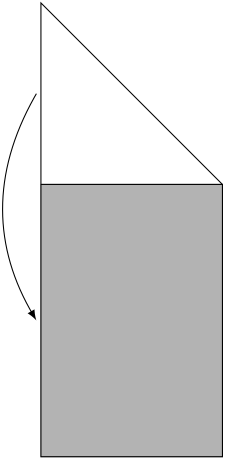
1. We consider all the fully summed variables

Strategy FSUD



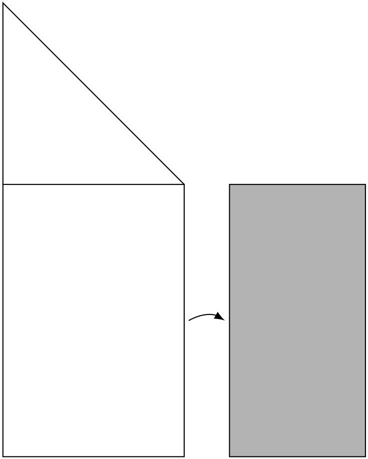
1. We consider all the fully summed variables
2. **F***actor* the entire diagonal block

Strategy FSUD



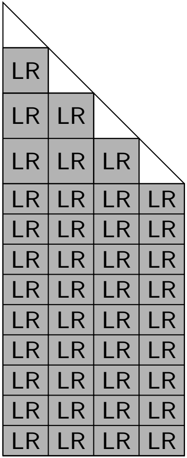
1. We consider all the fully summed variables
2. **F***actor* the entire diagonal block
3. **S***olve* operation on the off-diagonal block

Strategy FSUD



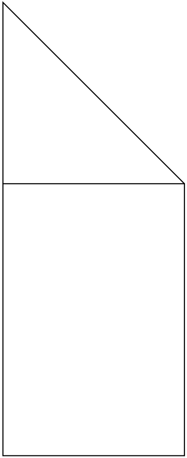
1. We consider all the fully summed variables
2. **F***actor* the entire diagonal block
3. **S***olve* operation on the off-diagonal block
4. **U***pdate* the Schur complement

Strategy FSUD



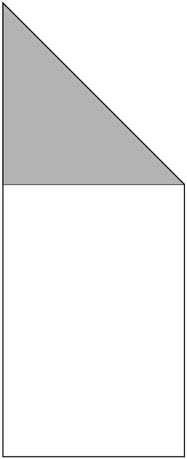
1. We consider all the fully summed variables
2. **F***actor* the entire diagonal block
3. **S***olve* operation on the off-diagonal block
4. **U***pdate* the Schur complement
5. **D***emote* each block of grouped variables within the factor

Strategy FSDU



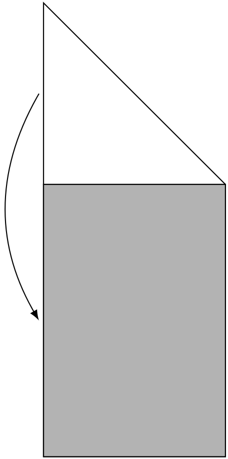
1. We process the entire fully summed variables block

Strategy FSDU



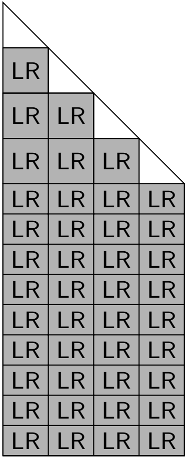
1. We process the entire fully summed variables block
2. **F***actor* the entire diagonal block

Strategy FSDU



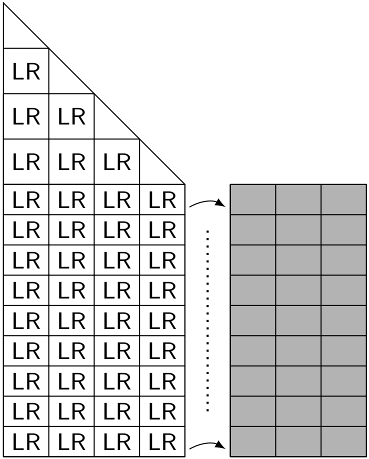
1. We process the entire fully summed variables block
2. **F***actor* the entire diagonal block
3. **S***olve* operation on the off-diagonal block

Strategy FSDU



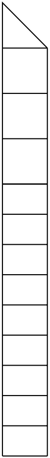
1. We process the entire fully summed variables block
2. **F***actor* the entire diagonal block
3. **S***olve* operation on the off-diagonal block
4. **D***emote* each block of grouped variables

Strategy FSDU

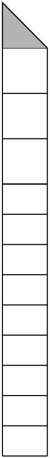


1. We process the entire fully summed variables block
2. **F***actor* the entire diagonal block
3. **S***olve* operation on the off-diagonal block
4. **D***emote* each block of grouped variables
5. *LR*-**U***pdate* the Schur complement blockwise

Strategy panel FSDU

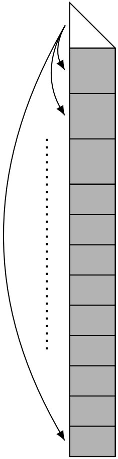


1. We process the fully summed variables block **panel***wise*



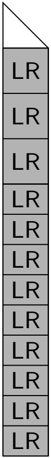
1. We process the fully summed variables block **panel**_wise_
2. **F**_actor_ the entire diagonal subblock

Strategy panel FSDU



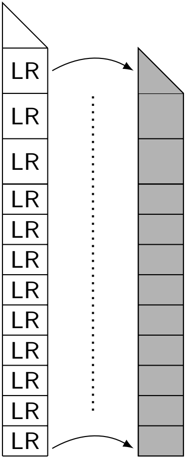
1. We process the fully summed variables block **panel***wise*
2. **F***actor* the entire diagonal subblock
3. **S***olve* operation on the off-diagonal subblocks

Strategy panel FSDU



1. We process the fully summed variables block **panel***wise*
2. **F***actor* the entire diagonal subblock
3. **S***olve* operation on the off-diagonal subblocks
4. **D***emote* each off-diagonal subblock

Strategy panel FSDU



1. We process the fully summed variables block **panel***wise*
2. **F***actor* the entire diagonal subblock
3. **S***olve* operation on the off-diagonal subblocks
4. **D***emote* each off-diagonal subblock
5. *LR*-**U***pdate* the trailing panels

Set of matrices

Matrix	N	NZ	Type	CSR
tpll01a_r6_t	66,053	295,947	thermic	6.4E-16
tpll01a_r7_t	263,173	1,181,707	thermic	8.6E-16
tpll01a_r8_t	1,050,629	4,722,699	thermic	1.3E-15
tpll01a_r6_m	132,106	1,117,719	mechanic	8.3E-16
tpll01a_r7_m	526,346	4,463,639	mechanic	1.8E-15
tpll01a_r8_m	2,101,258	17,840,151	mechanic	2.0E-15

- 2D problems
- thermo-mechanical simulations from Code_Aster (by EDF)
- different mesh refinements
- work still in progress on 3D problems
- Componentwise Scaled Residual $CSR = \frac{|\mathbf{b} - \mathbf{A}\bar{\mathbf{x}}|_i}{(|\mathbf{b}| + |\mathbf{A}|\ |\bar{\mathbf{x}}|)_i}$.

Strategy FSUD : results ($\varepsilon = 10^{-14}$)

Matrix	# of fronts	*L*	*MEMORY*	*OPS*	*CSR*
tpll01a_r6_t	6	10.3 %	49.9 %	–	9.2E-16
tpll01a_r7_t	22	17.4 %	35.7 %	–	9.1E-15
tpll01a_r8_t	111	28.2 %	29.4 %	–	3.6E-15
tpll01a_r6_m	100	22.2 %	58.0 %	–	1.5E-16
tpll01a_r7_m	423	45.1 %	59.9 %	–	1.8E-15
tpll01a_r8_m	468	40.1 %	41.7 %	–	1.0E-13

Features

- Focus on memory compression of the factorization
- No flop reduction during the factorization
- No error propagation within the factorization of the front
- Can be done "off-line" (solution phase, OOC)

Strategy FSDU : results ($\varepsilon = 10^{-14}$)

Matrix	# of fronts	*L*	*MEMORY*	*OPS*	*CSR*
tpll01a_r6_t	6	10.3 %	49.9 %	78.8 %	9.3E-16
tpll01a_r7_t	22	17.4 %	35.7 %	56.7 %	1.8E-15
tpll01a_r8_t	111	28.2 %	29.4 %	45.9 %	3.7E-14
tpll01a_r6_m	100	22.2 %	58.0 %	75.5 %	9.7E-15
tpll01a_r7_m	423	45.1 %	59.9 %	65.0 %	5.3E-15
tpll01a_r8_m	468	40.1 %	41.7 %	50.3 %	2.6E-13

- Memory compression for the storage of the factor
- More efficient update of the Schur complement thanks to the *LR-update* operation

Strategy panel FSDU : results ($\varepsilon = 10^{-14}$)

Matrix	# of fronts	*L*	*MEMORY*	*OPS*	*CSR*
tpll01a_r6_t	6	10.3 %	49.9 %	61.7 %	9.6E-16
tpll01a_r7_t	22	17.4 %	35.7 %	39.9 %	2.0E-15
tpll01a_r8_t	111	28.2 %	29.4 %	28.2 %	1.2E-14
tpll01a_r6_m	100	22.2 %	58.0 %	65.0 %	5.8E-15
tpll01a_r7_m	423	45.1 %	59.9 %	52.1 %	1.7E-15
tpll01a_r8_m	468	40.1 %	41.7 %	34.5 %	2.3E-13

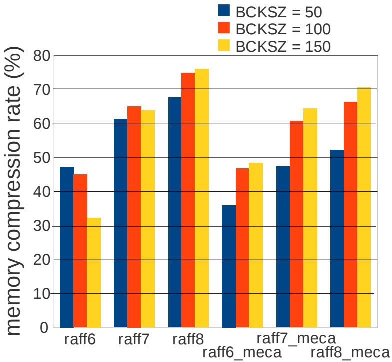
- Memory compression for the storage of the factor
- More efficient factorization due to *LR-updates* within the factor
- More efficient update of the Schur complement thanks to the *LR-update* operation

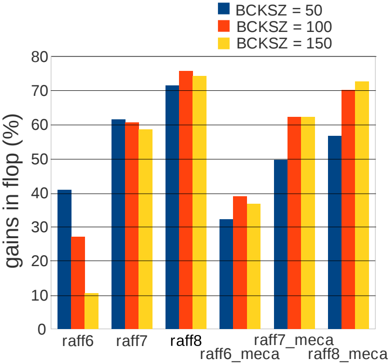
Comparison

	OPS (%)			ERROR				COND.
	FSUD	FSDU	panel FSDU	MUMPS	FSUD	FSDU	panel FSDU	
tpll01a_r6_t	–	78.8	61.7	6.4E-16	9.2E-16	9.3E-16	9.6E-16	4.0E+5
tpll01a_r7_t	–	56.7	39.9	8.6E-16	9.1E-15	1.8E-15	2.0E-15	1.7E+6
tpll01a_r8_t	–	45.9	28.2	1.3E-15	3.6E-15	3.7E-14	1.2E-14	6.9E+6
tpll01a_r6_m	–	75.5	65.0	8.3E-16	1.5E-16	9.7E-15	5.8E-15	7.4E+6
tpll01a_r7_m	–	65.0	52.1	1.8E-15	1.8E-15	5.3E-15	1.7E-15	3.3E+7
tpll01a_r8_m	–	50.3	34.5	2.0E-15	1.0E-13	2.6E-13	2.3E-13	1.5E+8

- not much error propagation from panel to panel
- Strategy panel FSDU is the most efficient
- need to process more fronts with the same efficiency

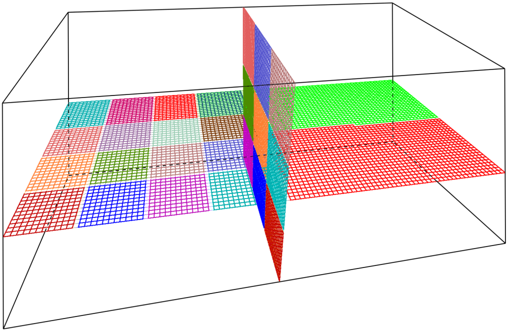
Influence of the block size





- stability for a large enough block size
- efficient block size depends on the front size
- will ease parallelism adaptation

3D target grouping (7-pts stencil, laplacian)



- hand-made separators (2 levels) and partitioning
- grouping results are close to this
- efficient on 2D separators
- problems with irregular separators

Error and accuracy

Local error on blocks : $\boxed{\frac{\|B - B_k\|_F}{\|B\|_F} \leq \varepsilon}$

Global error on solution : $\boxed{\sim \varepsilon}$

No propagation observed !

Summarizing

- → Strong link between structural and numerical aspects
- → Several gain spots

Summarizing

→ Strong link between structural and numerical aspects

→ Several gain spots

Two approaches

Summarizing

→ Strong link between structural and numerical aspects

→ Several gain spots

Two approaches

1 : PSEUDO-EXACT

- $\varepsilon \sim 10^{-16}$
- little accuracy lost
- typically used to accurately solve linear systems

Summarizing

→ Strong link between structural and numerical aspects

→ Several gain spots

Two approaches

1 : PSEUDO-EXACT

- $\varepsilon \sim 10^{-16}$
- little accuracy lost
- typically used to accurately solve linear systems

2 : APPROXIMATED

- $\varepsilon \gg 10^{-16}$
- typically used to compute preconditioners
- can replace mixed precision iterative refinement

Conclusion

- Efficient method for 2D problems
- Work still in progress for 3D problems
- An important step : the partitioning of the separator

Further works

- Pivoting, OOC, parallelism . . .
- Need to study the error propagation
- Theoretical work to have a better understanding of how the grouping should work
- 3D separator quality and partitioning

Questions

Thank you for your attention !

Any question ?