



AIRBUS



RÉGION
**Nouvelle-
Aquitaine**

Direct solution of larger coupled sparse/dense linear systems using low-rank compression on single-node multi-core machines in an industrial context

36th IEEE International Parallel and Distributed Processing symposium

Emmanuel Agullo, Marek Felšöci, Guillaume Sylvand

May 30 - June 3, 2022

Inria Bordeaux Sud-Ouest, France

Team-project CONCACE

Introduction

AIRBUS

- study the propagation of sound waves emitted by an aircraft
 - acoustic pollution reduction, prototype certification
- discrete model for numerical simulations
 - volume domain \mathbf{v} (jet flow)
 - Finite Elements Method (FEM) [10, 8]
 - surface domain \mathbf{s} (surface of the aircraft and the volume domain)
 - Boundary Elements Method (BEM) [5, 12]



An acoustic wave (blue arrow) emitted by the aircraft's engine, reflected on the wing and crossing the jet flow. Real-life case [11] (left) and a numerical model example (right).

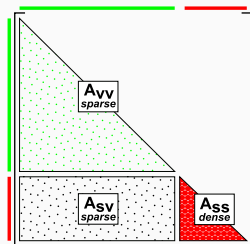
Problem

Global linear system coupling [6, 7] the FEM and the BEM unknowns:

$$\begin{bmatrix} A_{VV} & A_{SV}^T \\ A_{SV} & A_{SS} \end{bmatrix} \times \begin{bmatrix} x_V \\ x_S \end{bmatrix} = \begin{bmatrix} b_V \\ b_S \end{bmatrix}$$



- symmetric coefficient matrices:
 - sparse parts - volume domain v discretization with FEM (A_{VV}), surface/volume domain interaction (A_{SV})
 - a dense part - surface domain s discretization with BEM (A_{SS})
- finer model \rightarrow larger system
- direct solution using Schur complement [13]



Direct solution

Schur complement

- reduce the problem on boundaries \rightarrow simplify the system to solve

$$\begin{matrix} R_1 \\ R_2 \end{matrix} \begin{matrix} \color{green}{\rule{1cm}{2pt}} & \color{red}{\rule{1cm}{2pt}} \\ \left[\begin{array}{cc} A_{VV} & A_{SV}^T \\ A_{SV} & A_{SS} \end{array} \right] \end{matrix} \times \begin{bmatrix} x_V \\ x_S \end{bmatrix} = \begin{bmatrix} b_V \\ b_S \end{bmatrix}$$

Computation steps

- eliminate x_V from the second equation \rightarrow Schur complement S

$$\begin{matrix} R_1 \\ R_2 \leftarrow R_2 - A_{SV} A_{VV}^{-1} \times R_1 \end{matrix} \begin{bmatrix} A_{VV} & A_{SV}^T \\ 0 & \underbrace{A_{SS} - A_{SV} A_{VV}^{-1} A_{SV}^T}_S \end{bmatrix} \times \begin{bmatrix} x_V \\ x_S \end{bmatrix} = \begin{bmatrix} b_V \\ b_S - A_{SV} A_{VV}^{-1} b_V \end{bmatrix}$$

- solve the reduced Schur complement system

$$S x_S = b_S - A_{SV} A_{VV}^{-1} b_V$$

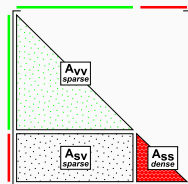
- determine x_V using x_S

$$x_V = A_{VV}^{-1} (b_V - A_{SV}^T x_S)$$

Numerical computation

Properties of the input linear system

- A_{VV} and A_{SS} are symmetric
- A_{VV} and A_{SV} are sparse



Initial state of A

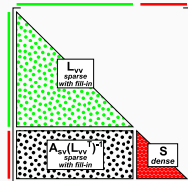
Ideal computation of $S = A_{SS} - A_{SV}A_{VV}^{-1}A_{VS}$

- symmetric factorization of $A_{VV} \rightarrow L_{VV}L_{VV}^T$: fill-in

$$S = A_{SS} - A_{SV}(L_{VV}L_{VV}^T)^{-1}A_{SV}^T$$

- computation of the Schur complement

$$S = A_{SS} - \underbrace{(A_{SV}(L_{VV}^T)^{-1})}_{\text{triangular solve}} \underbrace{(A_{SV}(L_{VV}^T)^{-1})^T}_{\text{implicitly known}}$$



A after computing S

Implementation

- coupling of a **sparse** direct and a **dense** direct solver
 - fully-featured community solvers implement appealing functionalities we want to take advantage of
 - **low-rank compression**, out-of-core computation, distributed memory parallelism
- two different schemes depending on the choice of **sparse** solver's building blocks
 - *baseline* coupling and *advanced* coupling

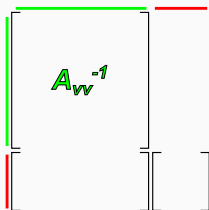
baseline coupling

- separate A_{VV} , A_{SV} and A_{SS}
- *sparse facto., sparse solve*
- *dense facto., dense solve*

Vanilla solver couplings

baseline coupling

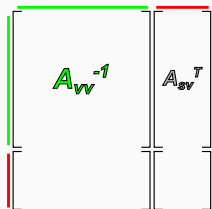
- separate A_{VV} , A_{SV} and A_{SS}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



Vanilla solver couplings

baseline coupling

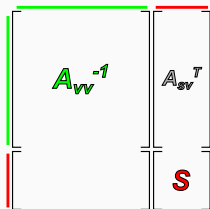
- separate A_{VV} , A_{SV} and A_{SS}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



Vanilla solver couplings

baseline coupling

- separate A_{VV} , A_{SV} and A_{SS}
- *sparse facto., sparse solve*
- *dense facto., dense solve*



Vanilla solver couplings

baseline coupling

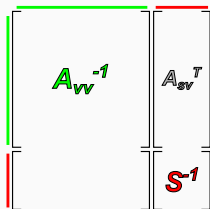
- separate A_{VV} , A_{SV} and A_{SS}
- *sparse facto., sparse solve*
- *dense facto., dense solve*

$$\begin{bmatrix} A_{VV}^{-1} & A_{SV}^T \\ & S^{-1} \end{bmatrix}$$

Vanilla solver couplings

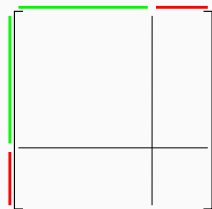
baseline coupling

- separate A_{VV} , A_{SV} and A_{SS}
- *sparse facto., sparse solve*
- *dense facto., dense solve*



advanced coupling

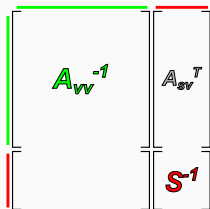
- A as a whole
- *sparse facto.+Schur*
- *dense facto., dense solve*



Vanilla solver couplings

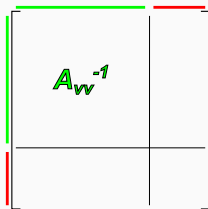
baseline coupling

- separate A_{VV} , A_{SV} and A_{SS}
- *sparse facto., sparse solve*
- *dense facto., dense solve*



advanced coupling

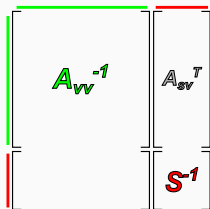
- A as a whole
- *sparse facto.+Schur*
- *dense facto., dense solve*



Vanilla solver couplings

baseline coupling

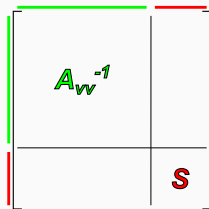
- separate A_{VV} , A_{SV} and A_{SS}
- *sparse facto., sparse solve*
- *dense facto., dense solve*



A block matrix diagram for baseline coupling. It consists of two columns. The left column contains a green A_{VV}^{-1} block. The right column contains a red A_{SV}^T block above a red S^{-1} block. A green bracket is on the left of the top row, and a red bracket is on the left of the bottom row.

advanced coupling

- A as a whole
- *sparse facto.+Schur*
- *dense facto., dense solve*



A block matrix diagram for advanced coupling. It consists of two columns. The left column contains a green A_{VV}^{-1} block. The right column contains a red S block. A green bracket is on the left of the top row, and a red bracket is on the left of the bottom row.

Vanilla solver couplings

baseline coupling

- separate A_{VV} , A_{SV} and A_{SS}
- *sparse facto., sparse solve*
- *dense facto., dense solve*

$$\begin{bmatrix} A_{VV}^{-1} & A_{SV}^T \\ S^{-1} \end{bmatrix}$$

advanced coupling

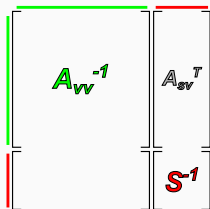
- A as a whole
- *sparse facto.+Schur*
- *dense facto., dense solve*

$$\begin{bmatrix} A_{VV}^{-1} & \\ S^{-1} \end{bmatrix}$$

Vanilla solver couplings

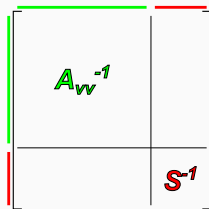
baseline coupling

- separate A_{VV} , A_{SV} and A_{SS}
- *sparse facto., sparse solve*
- *dense facto., dense solve*



advanced coupling

- A as a whole
- *sparse facto.+Schur*
- *dense facto., dense solve*

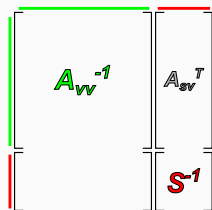


- S non-compressed, dense
- A_{SV}^T explicitly stored, dense

Vanilla solver couplings

baseline coupling

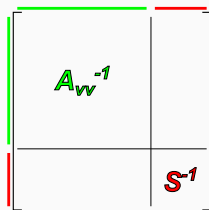
- separate A_{VV} , A_{SV} and A_{SS}
- *sparse facto., sparse solve*
- *dense facto., dense solve*



- S non-compressed, dense
- A_{SV}^T explicitly stored, dense

advanced coupling

- A as a whole
- *sparse facto.+Schur*
- *dense facto., dense solve*



- S non-compressed, dense

Contribution

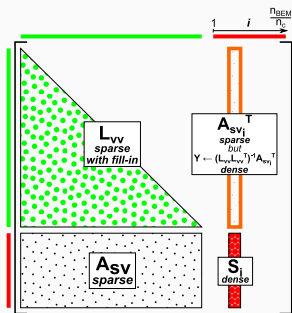
Coping with limitations

- keep using fully-featured well optimized community solvers despite limitations in their API
- two new algorithms allowing for low-rank compression of $S \rightarrow$ processing of larger coupled systems
 1. **multi-solve** based on the *baseline* coupling
 2. **multi-factorization** based on the *advanced* coupling

Multi-solve

$$S_i = A_{ss_i} - A_{sv} \overbrace{(L_{vv}L_{vv}^T)^{-1}A_{sv_i}^T}^{\text{solve} \rightarrow Y_i}$$

- 1 *sparse facto.* of the **green** matrix (symmetric)
- plenty of *sparse solve* involving the **orange** blocks (result is dense)

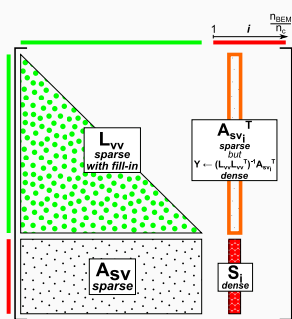


WITHOUT compression

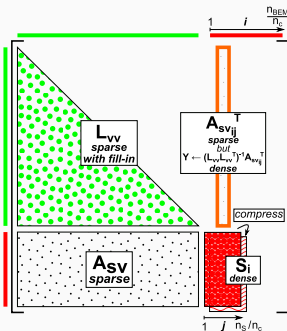
Multi-solve

$$S_i = A_{ss_i} - A_{sv} \overbrace{(L_{vv}L_{vv}^T)^{-1}A_{sv_i}^T}^{\text{solve} \rightarrow Y_i}$$

- 1 *sparse facto.* of the **green** matrix (symmetric)
- plenty of *sparse solve* involving the **orange** blocks (result is dense)



WITHOUT compression

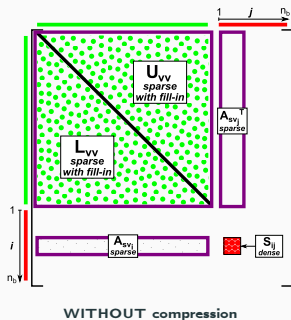


WITH compression

Multi-factorization

$$S_{ij} = A_{ssij} - \overbrace{A_{sv_i} (L_{vv} U_{vv})^{-1} A_{sv_j}^T}^{\text{used with Schur API}}$$

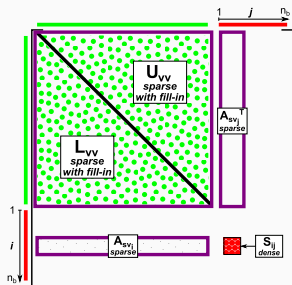
- multiple *sparse facto.* + *Schur* of the **violet** matrix (non-symmetric)
- computation of the Schur complement blocks via API



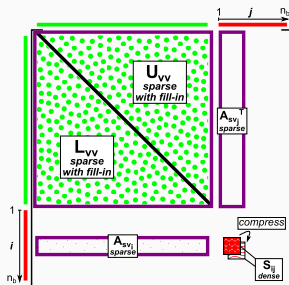
Multi-factorization

$$S_{ij} = A_{ss_{ij}} - \overbrace{A_{sv_i} (L_{vv} U_{vv})^{-1} A_{sv_j}^T}^{\text{used with Schur API}}$$

- multiple *sparse facto.* + *Schur* of the **violet** matrix (non-symmetric)
- computation of the Schur complement blocks via API



WITHOUT compression

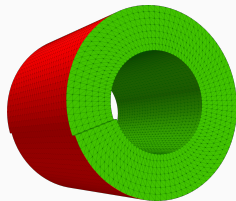


WITH compression

Experimental evaluation

Academic test case

- short pipe (length: 2 m, radius: 4 m) [2]
- systems close enough to real-life models
- FEM/BEM mesh (**v** and **s** parts)



A short pipe (20,000 unknowns)

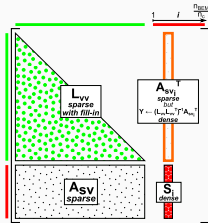
Configuration

- PlaFRIM [1], 24-core *miriel* nodes
 - 128 GiB of RAM
- precision parameter ϵ set to 10^{-3}
- **sparse** solver with **low-rank compression always on**
- **out-of-core disabled**

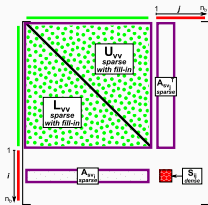
Implementation

- **sparse** solver
 - MUMPS [3]
- **dense** solvers
 - SPIDO (non-compressed S)
 - HMAT (compressed S) [9]

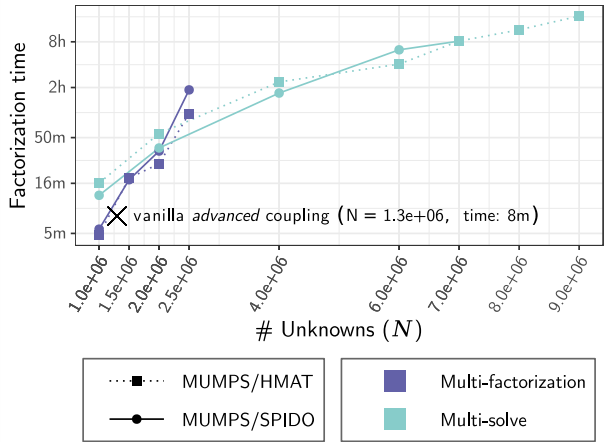
Solving larger systems



multi-solve

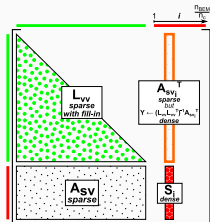


multi-factorization

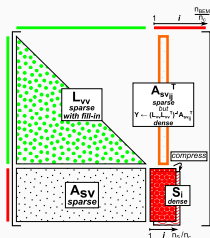


Best computation times of multi-solve and multi-factorization for both solver couplings MUMPS/HMAT and MUMPS/SPIDO. Parallel runs using 24 threads on single miriel node.

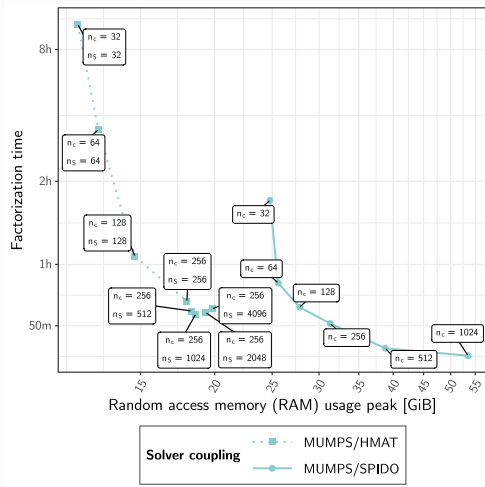
Performance-memory trade-off of multi-solve



multi-solve
(non-compressed)

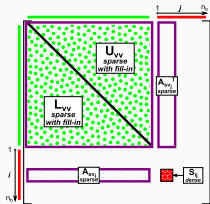


multi-solve
(compressed)

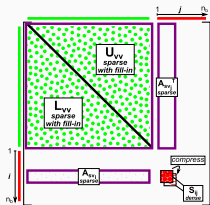


Comparison between the non-compressed and compressed multi-solve for the MUMPS/SPIDO and MUMPS/HMAT couplings on a coupled FEM/BEM system with 2,000,000 unknowns for varying n_c and n_s .

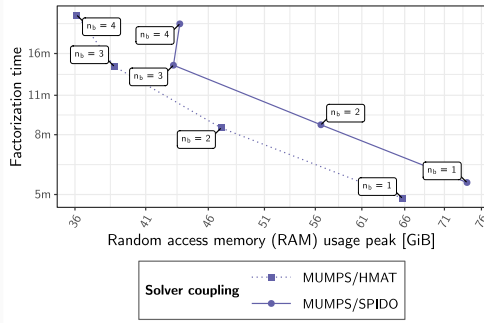
Performance-memory trade-off of multi-factorization



multi-factorization
(non-compressed)



multi-factorization
(compressed)

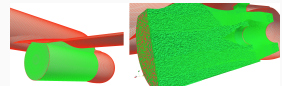


Comparison between the non-compressed and compressed multi-factorization for the MUMPS/SPIDO and MUMPS/HMAT couplings on a coupled FEM/BEM system with 1,000,000 unknowns for varying n_b .

Industrial application

Industrial test case

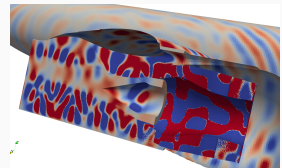
- 2,259,468 unknowns
 - 2,090,638 in the **v** part
 - 168,830 in the **s** part
- 32-core machine with 384 GiB of RAM



Model FEM/BEM mesh

Results

Algorithm	RAM (GiB)	Time
vanilla <i>advanced</i> coupling	>384	N/A
multi-solve (non-compressed S)	224	15h
multi-factorization (non-compressed S)	275	8h
multi-solve (compressed S)	35	9h
multi-factorization (compressed S)	137	51m



Computed acoustic pressure

Conclusion

Contribution

- two algorithms allowing us to:
 - benefit from the most advanced functionalities of fully-featured solvers
 - process larger systems compared to vanilla couplings
 - 9M ([multi-solve](#)) and 2.5M ([multi-factorization](#)) vs. 1.3M on a single 24-core, 128 GiB RAM workstation
- confirm the advantage of compressing the Schur complement
- validate the algorithms on a real-life industrial case

Limitations

- **multi-solve**: explicit storage of an intermediate result in a non-compressed dense matrix
- **multi-factorization**: superfluous re-factorizations of the **sparse** submatrix A_{VV}

Ongoing work

- extension to out-of-core and distributed memory cases
- task-based solver coupling based on the StarPU runtime [4]

Acknowledgement

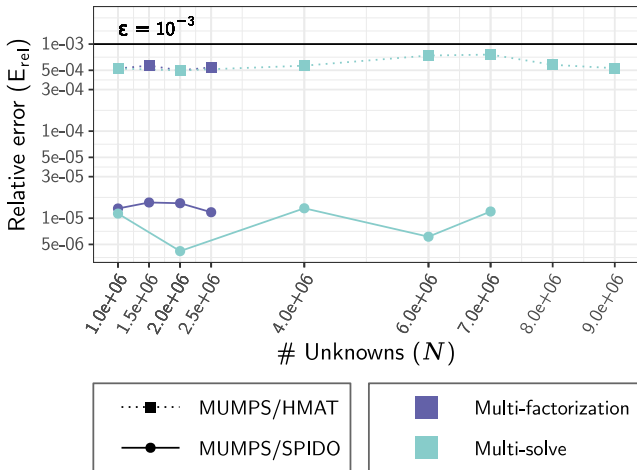
- Projet Région Nouvelle-Aquitaine 2018-1R50119
« HPC scalable ecosystem »
- PlaFRIM experimental testbed [1] (supported by Inria, CNRS - LABRI and IMB, Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine)

Thank you for attending!

- [1] *PlaFRIM: Plateforme fédérative pour la recherche en informatique et mathématiques*.
<https://plafrim.fr/>.
- [2] *test_FEMBEM, a simple application for testing dense and sparse solvers with pseudo-FEM or pseudo-BEM matrices*.
https://gitlab.inria.fr/solverstack/test_fembem.
- [3] P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent, *MUMPS multifrontal massively parallel solver version 2.0*, (1998).
- [4] C. Augonnet, S. Thibault, and R. Namyst, *StarPU: a Runtime System for Scheduling Tasks over Accelerator-Based Multicore Machines*, Rapport de recherche RR-7240, INRIA, Mar. 2010.
- [5] P. K. Banerjee and R. Butterfield, *Boundary element methods in engineering science*, vol. 17, McGraw-Hill London, 1981.
- [6] F. Casenave, *Méthodes de réduction de modèles appliquées à des problèmes d'aéroacoustique résolus par équations intégrales*, PhD thesis, Université Paris-Est, 2013.
- [7] F. Casenave, A. Ern, and G. Sylvand, *Coupled BEM-FEM for the convected Helmholtz equation with non-uniform flow in a bounded domain*, *Journal of Computational Physics*, 257 (2014), pp. 627–644.
- [8] A. Ern and J.-L. Guermond, *Theory and practice of finite elements*, vol. 159, Springer Science & Business Media, 2013.
- [9] B. Lizé, *Résolution Directe Rapide pour les Éléments Finis de Frontière en Électromagnétisme et Acoustique : \mathcal{H} -Matrices. Parallélisme et Applications Industrielles*, PhD thesis, Université Paris 13, 2014.
- [10] P. Raviart and J. Thomas, *A mixed finite element method for 2-nd order elliptic problems*, in *Mathematical Aspects of Finite Element Methods*, I. Galligani and E. Magenes, eds., vol. 606 of *Lecture Notes in Mathematics*, Springer Berlin Heidelberg, 1977, pp. 292–315.
- [11] Sebaso, *Jet engine airflow during take-off*.
https://commons.wikimedia.org/wiki/File:20140308-Jet_engine_airflow_during_take-off.jpg.
- [12] A. Wang, N. Vlahopoulos, and K. Wu, *Development of an energy boundary element formulation for computing high-frequency sound radiation from incoherent intensity boundary conditions*, *Journal of Sound and Vibration*, 278 (2004), pp. 413–436.
- [13] F. Zhang, *The Schur complement and its applications*, vol. 4, Springer Science & Business Media, 2006.

Appendix

Numerical assessment

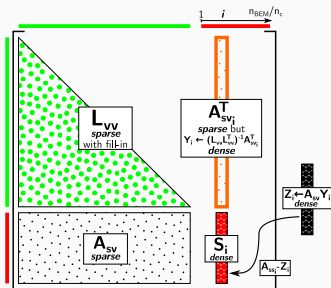


Relative error E_{rel} for the runs of **multi-solve** and **multi-factorization** having the best execution times and for both solver couplings MUMPS/HMAT and MUMPS/SPIDO. Parallel runs using 24 threads on single *miriel* node.

Multi-solve

$$S_i = A_{ss_i} - \underbrace{A_{sv} \overbrace{(L_{vv} L_{vv}^T)^{-1} A_{sv}^T}^{\text{solve} \rightarrow Y_i}}_{Z_i}$$

- 1 *sparse facto.* of the **green** matrix (symmetric)
- plenty of *sparse solve* involving the **orange** blocks (result is dense)

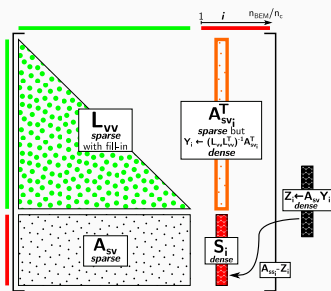


WITHOUT compression

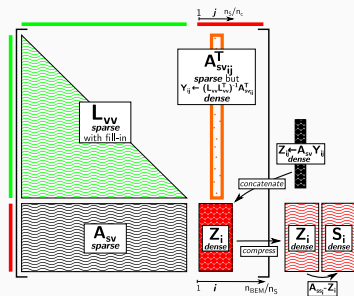
Multi-solve

$$S_i = A_{ss_i} - \underbrace{A_{sv} \left(L_{vv} L_{vv}^T \right)^{-1} A_{sv}^T}_{Z_i} Y_i$$

- 1 *sparse facto.* of the **green** matrix (symmetric)
- plenty of *sparse solve* involving the **orange** blocks (result is dense)



WITHOUT compression



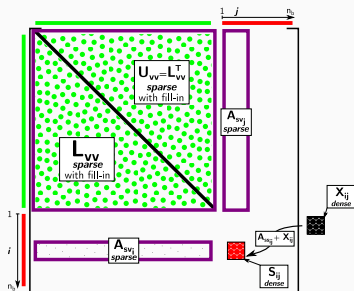
WITH compression

Multi-factorization

used with Schur API $\rightarrow X_{ij}$

$$S_{ij} = A_{ss_{ij}} - \overbrace{A_{sv_i} (L_{vv} U_{vv})^{-1} A_{sv_j}^T}^{X_{ij}}$$

- multiple *sparse facto. + Schur* of the **violet** matrix (non-symmetric)
- computation of the Schur complement blocks via API



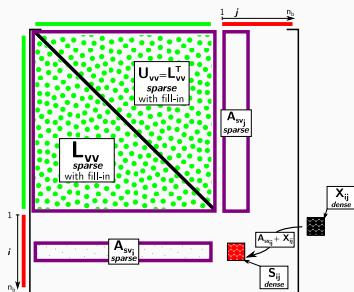
WITHOUT compression

Multi-factorization

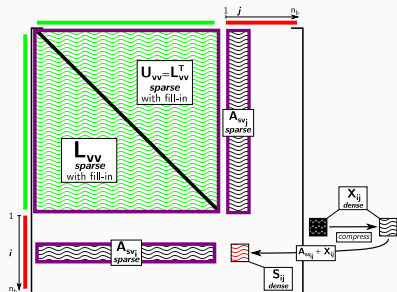
used with Schur API $\rightarrow X_{ij}$

$$S_{ij} = A_{ss_{ij}} - \overbrace{A_{sv_i} (L_{vv} U_{vv})^{-1} A_{sv_j}^T}^{X_{ij}}$$

- multiple *sparse facto.* + Schur of the **violet** matrix (non-symmetric)
- computation of the Schur complement blocks via API



WITHOUT compression



WITH compression