Direct solution of larger coupled sparse/dense FEM/BEM linear systems using low-rank compression

Sparse Days

Emmanuel Agullo, Marek Felšöci, Guillaume Sylvand

June 20 - 22, 2022

Inria Bordeaux Sud-Ouest, France
Team-project CONCACE

Industrial context

AIRBUS

- study the propagation of sound waves emitted by an aircraft
  - acoustic pollution reduction, prototype certification
- discrete model for numerical simulations
  - volume domain v (jet flow)
    - Finite Elements Method (FEM) [11, 9]
  - surface domain s (surface of the aircraft and the volume domain)
    - Boundary Elements Method (BEM) [6, 13]

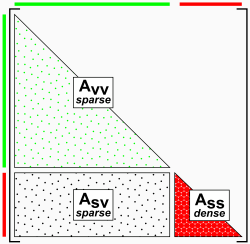
An acoustic wave (blue arrow) emitted by the aircraft's engine, reflected on the wing and crossing the jet flow. Real-life case [12] (left) and a numerical model example (right).

Problem

Global linear system coupling [7, 8] 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 [14]



Direct solution

Schur complement

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

$$\begin{array}{c} R_1 \\ R_2 \end{array} \begin{bmatrix} A_{vv} & A_{sv}^T \\ A_{sv} & A_{ss} \end{bmatrix} \times \begin{bmatrix} x_v \\ x_s \end{bmatrix} = \begin{bmatrix} b_v \\ b_s \end{bmatrix}$$

Computation steps

1. eliminate x_v from the second equation $\rightarrow$ Schur complement S

$$\begin{array}{c} R_1 \\ R_2 \leftarrow R_2 - A_{sv} A_{vv}^{-1} \times R_1 \end{array} \begin{bmatrix} A_{vv} & A_{sv}^T \\ 0 & \underbrace{A_{ss} - A_{sv} A_{vv}^{-1} A_{sv}^T}_{S} \end{bmatrix} \times \begin{bmatrix} x_v \\ x_s \end{bmatrix} = \begin{bmatrix} b_v \\ b_s - A_{sv} A_{vv}^{-1} b_v \end{bmatrix}$$

2. solve the reduced Schur complement system

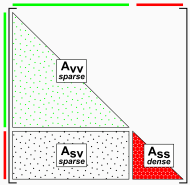
$$S x_s = b_s - A_{sv} A_{vv}^{-1} b_v$$

3. determine x_v using x_s

$$x_v = A_{vv}^{-1}(b_v - A_{sv}^T x_s)$$

Properties of the input linear system

- A_{vv} and A_{ss} are symmetric
- A_{vv} and A_{sv} are sparse



Initial state of A

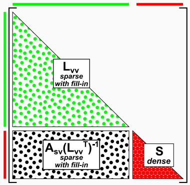
Ideal computation of $S = A_{ss} - A_{sv} A_{vv}^{-1} A_{vs}$

- 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 with appealing functionalities
    - low-rank compression, out-of-core, distributed memory parallelism
- two different schemes depending on the way of using the building blocks of the sparse solver
  - *baseline* coupling
  - *advanced* coupling

Vanilla solver couplings

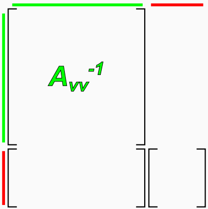
baseline **coupling**

- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*

Vanilla solver couplings

***baseline* coupling**

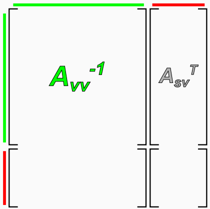
- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



Vanilla solver couplings

***baseline* coupling**

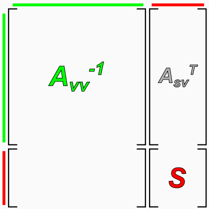
- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



Vanilla solver couplings

***baseline* coupling**

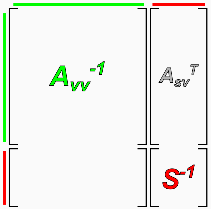
- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



Vanilla solver couplings

***baseline* coupling**

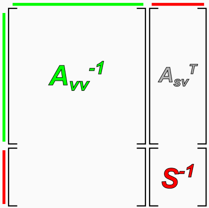
- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



Vanilla solver couplings

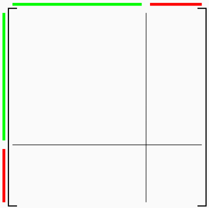
***baseline* coupling**

- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



***advanced* coupling**

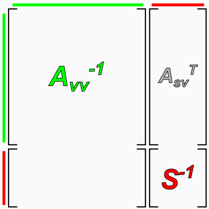
- A as a whole
- *sparse facto.+Schur*
- *dense facto.*, *dense solve*



Vanilla solver couplings

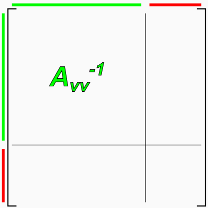
***baseline* coupling**

- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



***advanced* coupling**

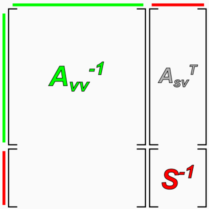
- A as a whole
- *sparse facto.+Schur*
- *dense facto.*, *dense solve*



Vanilla solver couplings

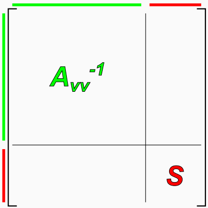
baseline coupling

- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto., sparse solve*
- *dense facto., dense solve*



advanced coupling

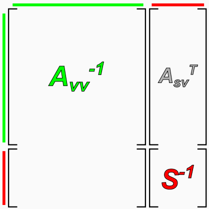
- A as a whole
- *sparse facto.+Schur*
- *dense facto., dense solve*



Vanilla solver couplings

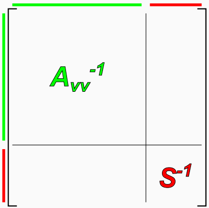
***baseline* coupling**

- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



***advanced* coupling**

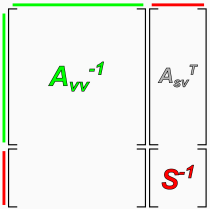
- A as a whole
- *sparse facto.+Schur*
- *dense facto.*, *dense solve*



Vanilla solver couplings

***baseline* coupling**

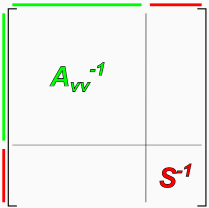
- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto., sparse solve*
- *dense facto., dense solve*



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

***advanced* coupling**

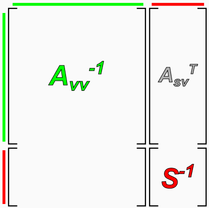
- A as a whole
- *sparse facto.+Schur*
- *dense facto., dense solve*



Vanilla solver couplings

baseline **coupling**

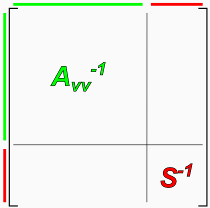
- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



- 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

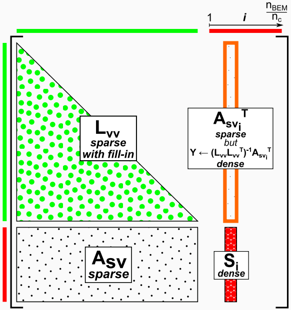
Coping with limitations

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

Multi-solve

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

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

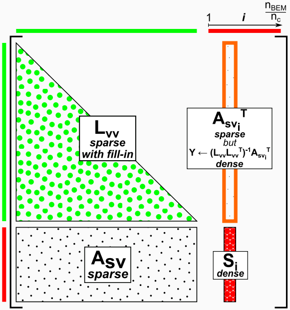


WITHOUT compression

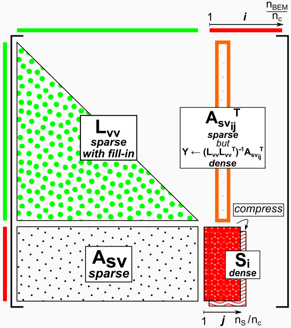
Multi-solve

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

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



WITHOUT compression

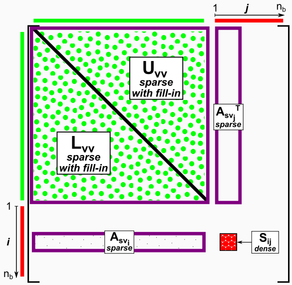


WITH compression

Multi-factorization

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

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

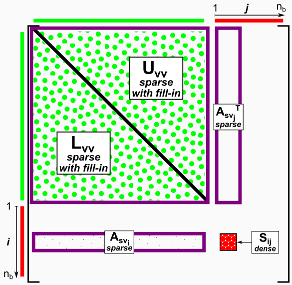


WITHOUT compression

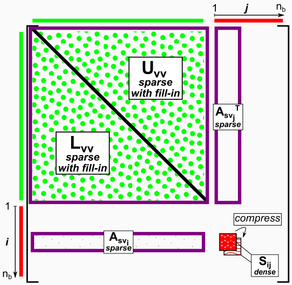
Multi-factorization

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

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



WITHOUT compression

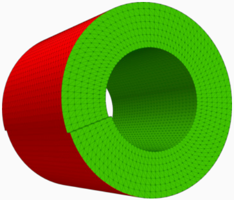


WITH compression

Experimental evaluation

Academic test case

- short pipe (length: 2 m, radius: 4 m) [3]
- systems close enough to real-life models
- FEM/BEM mesh (v and s parts)



A short pipe (20,000 unknowns)

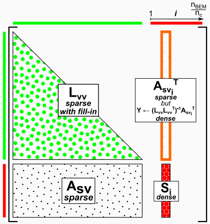
Configuration

- 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

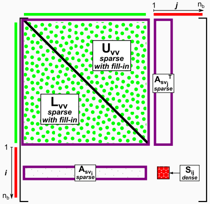
Implementation

- sparse solver
  - MUMPS [4]
- dense solvers
  - SPIDO (non-compressed S)
  - HMAT (compressed S) [10]

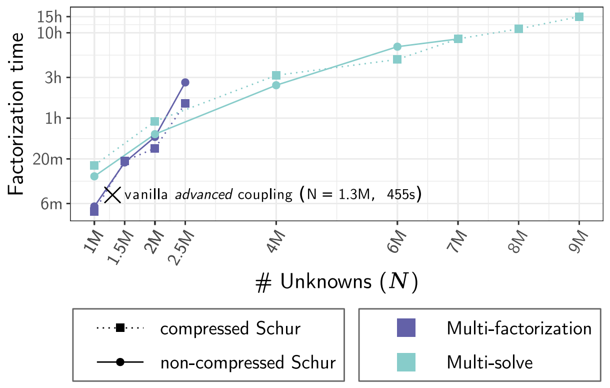
Solving larger systems



multi-solve

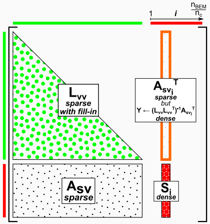


multi-factorization

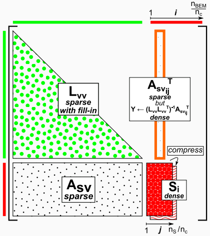


Best computation times of multi-solve and multi-factorization for both solver couplings, MUMPS/HMAT (compressed Schur) and MUMPS/SPIDO (non-compressed Schur). Parallel runs using 24 threads on single *miriel* node.

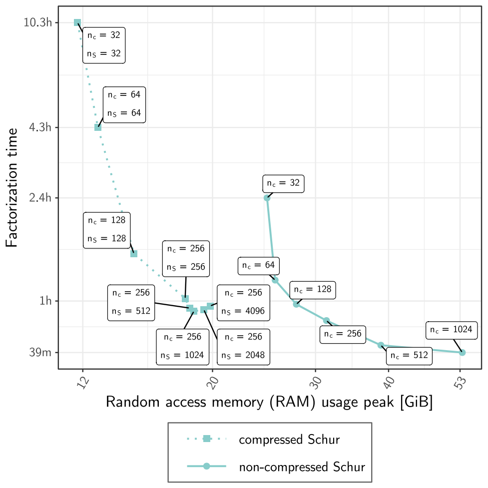
Performance-memory trade-off of multi-solve



multi-solve
(non-compressed)

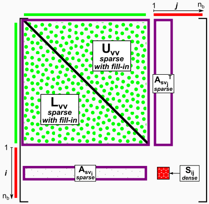


multi-solve
(compressed)

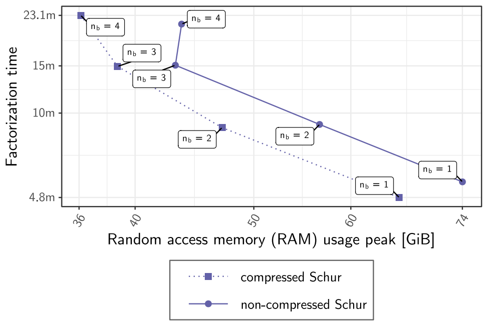


Comparison between the non-compressed and compressed multi-solve for the MUMPS/HMAT (compressed Schur) and the MUMPS/SPIDO (non-compressed Schur) couplings on a coupled FEM/BEM system with 2,000,000 unknowns for varying n_c and n_S.

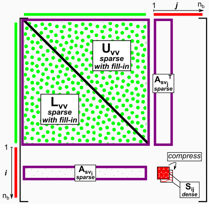
Performance-memory trade-off of multi-factorization



multi-factorization
(non-compressed)



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

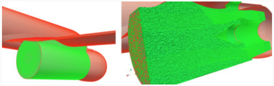


multi-factorization
(compressed)

Industrial application

Industrial test case

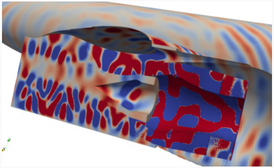
- 2,259,468 unknowns
  - 2,090,638 in the v part
  - 168,830 in the s part
- 32-core machine with 384 GiB of RAM



Model FEM/BEM mesh

Results

Algorithm	RAM (GiB)	Time
vanilla *advanced* 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

Summary

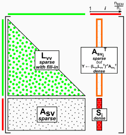
Contribution

- 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
    - industrial case impossible to run before on a single 32-core, 384 GiB RAM workstation
- confirm the advantage of compressing the Schur complement
- validate the algorithms on a real-life industrial case

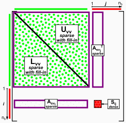
Ongoing work

- extension to out-of-core and distributed memory cases

Towards ideal implementation



multi-solve



multi-factorization

Main limitations

- multi-solve - explicit storage of dense orange blocks
- multi-factorization - re-factorizations of the green matrix

Collaboration with A. Buttari (IRIT/ENSEEIHT)

- coupling of task based direct solvers
  - sparse: `qr_mumps` [2]
    - no compression, no distributed memory parallelism
  - dense: HMAT
  - relying on the StarPU runtime [5]
    - built-in out-of-core capability
- S is never assembled entirely in memory
- dense solver can start working without waiting for S to be fully assembled

Thank you for attending!

Looking for a post-doctoral research position
(Ph.D. defense expected in January 2023)

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)

References i

[1] *PlaFRIM: Plateforme fédérative pour la recherche en informatique et mathématiques*.
`https://plafrim.fr/`.

[2] *qr_mumps, a software package for the solution of sparse, linear systems on multicore computers*.
`http://buttari.perso.enseeiht.fr/qr_mumps/`.

[3] *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`.

[4] P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent, *MUMPS multifrontal massively parallel solver version 2.0*, (1998).

[5] 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.

[6] P. K. Banerjee and R. Butterfield, *Boundary element methods in engineering science*, vol. 17, McGraw-Hill London, 1981.

[7] F. Casenave, *Méthodes de réduction de modèles appliquées à des problèmes d'aéroacoustique résolus par équations intégrales*, PhD thesis, Université Paris-Est, 2013.

[8] 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.

[9] A. Ern and J.-L. Guermond, *Theory and practice of finite elements*, vol. 159, Springer Science & Business Media, 2013.

[10] 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.

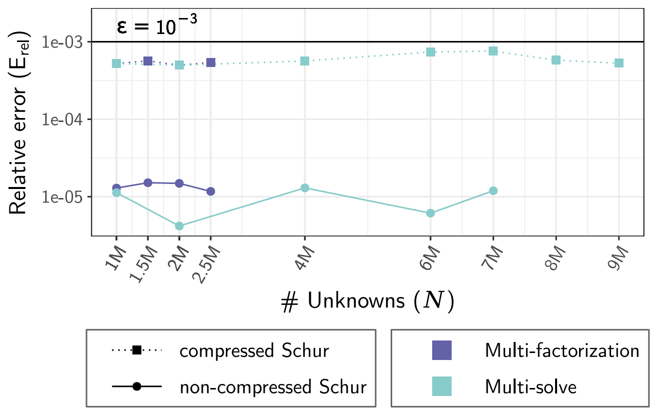
[11] P. Raviart and J. Thomas, *A mixed finite element method for 2-nd order elliptic problems*, in Mathematical Aspects of Finite Element Methods, I. Galligani and E. Magenes, eds., vol. 606 of Lecture Notes in Mathematics, Springer Berlin Heidelberg, 1977, pp. 292–315.

[12] Sebaso, *Jet engine airflow during take-off*. `https://commons.wikimedia.org/wiki/File:20140308-Jet_engine_airflow_during_take-off.jpg`.

[13] 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.

[14] F. Zhang, *The Schur complement and its applications*, vol. 4, Springer Science & Business Media, 2006.

Numerical assessment

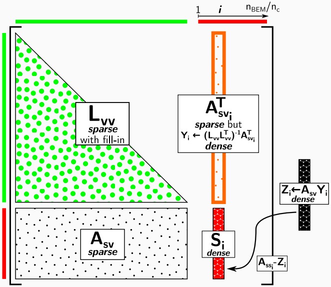


Relative error E_{rel} for the runs of multi-solve and multi-factorization having the best execution times and for both solver couplings, MUMPS/HMAT (compressed Schur) and MUMPS/SPIDO (non-compressed Schur). Parallel runs using 24 threads on single *miriel* node.

Multi-solve

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

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

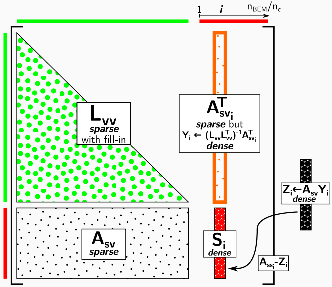


WITHOUT compression

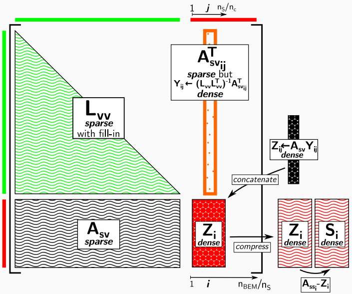
Multi-solve

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

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



WITHOUT compression

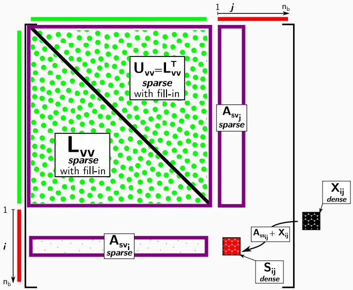


WITH compression

Multi-factorization

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

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

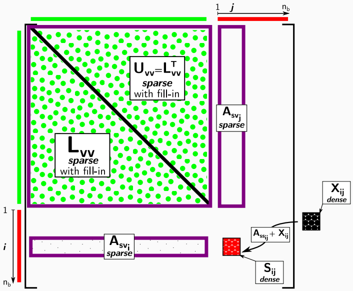


WITHOUT compression

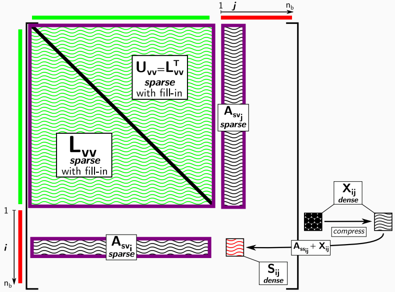
Multi-factorization

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

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



WITHOUT compression



WITH compression