A preliminary comparative study of solvers for coupled FEM/BEM linear systems in a reproducible environment

Emmanuel Agullo, Marek Felšöci, Guillaume Sylvand

December 8, 2020

Inria Bordeaux Sud-Ouest
Team-project HiePACS

Context

Fast solvers for high-frequency aeroacoustics

- Ph.D. thesis co-funded by Airbus and Région Nouvelle-Aquitaine
  - supervised by Guillaume Sylvand [1] and Emmanuel Agullo [2]
  - continuation of the thesis of Aurélien Falco [6]
- industrial context of Airbus
  - study of the propagation of sound waves emitted by an aircraft
    $\rightarrow$ reduction of acoustic pollution, certification of prototypes

Figure 1: **Jet flow emitted by an Airbus A319-112 at take-off (Sebaso, Wikimedia Commons, CC BY-SA 4.0)**

[1] Airbus Central R&T / Inria Bordeaux Sud-Ouest
[2] Inria Bordeaux Sud-Ouest

Towards a numerical model

Model representing the aircraft's surface and the emitted jet flow:

1. continuous physical model based on partial differential equations
2. discrete numerical model
   - volume domain v (jet flow) discretized using Finite Elements Method (FEM)
   - surface domain s (surface of the aircraft and the volume domain) discretized using Boundary Elements Method (BEM)

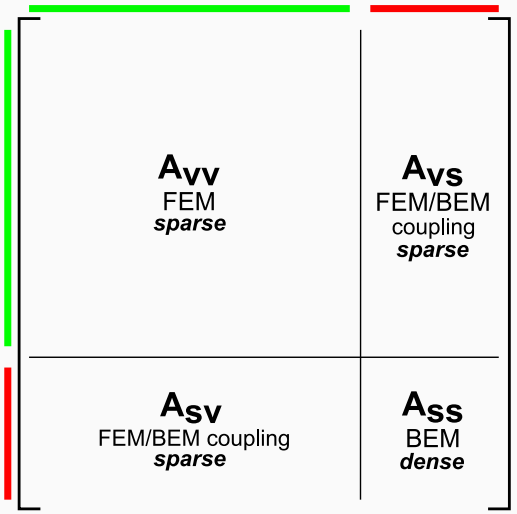
Figure 2: **Example of a FEM/BEM discrete model. The red mesh corresponds to the BEM discretization and the green mesh to the FEM discretization.**

FEM/BEM coupling

A global linear system coupling the unknowns associated with the formulation of both FEM and BEM:

- coefficient matrices composed of
  - sparse parts - discretization of the volume domain v using FEM (A_{vv}), surface/volume domain interaction (A_{vs}, A_{sv})
  - a dense part - discretization of the surface domain s using BEM (A_{ss})

$$\begin{bmatrix} A_{vv} & A_{vs} \\ A_{sv} & A_{ss} \end{bmatrix} \times \begin{bmatrix} x_v \\ x_s \end{bmatrix} = \begin{bmatrix} b_v \\ b_s \end{bmatrix}$$



- can be solved using direct or iterative methods

Direct solution

Using the Schur complement [8]

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

$$\begin{matrix} R_1 \\ R_2 \end{matrix} \begin{bmatrix} A_{vv} & A_{vs} \\ A_{sv} & A_{ss} \end{bmatrix} \times \begin{bmatrix} x_v \\ x_s \end{bmatrix} = \begin{bmatrix} b_v \\ b_s \end{bmatrix}$$

Main solution steps

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

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

2. solve the reduced Schur complement system

$$\underbrace{\left(A_{ss} - A_{sv} A_{vv}^{-1} A_{vs}\right)}_{S} x_s = b_s - A_{sv} A_{vv}^{-1} b_v$$

Fill-in

Coefficient matrix is symmetric

- A_{vv} and A_{ss} are symmetric and $A_{vs} = A_{sv}^T$
- reduces the amount of required storage space

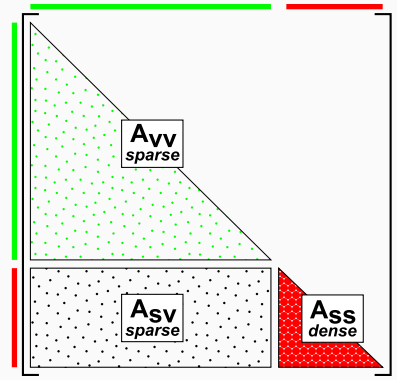


Figure 3: A **BEFORE the Schur complement computation**

Computation of the Schur complement S

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

- LL^T factorization of $A_{vv} \rightarrow L_{vv} L_{vv}^T$
- substitution of A_{vs} by A_{sv}^T

$$S = A_{ss} - A_{sv}(L_{vv} L_{vv}^T)^{-1} A_{sv}^T$$

$$S = A_{ss} - \underbrace{(A_{sv}(L_{vv}^T)^{-1})}_{\text{triangular solve}} \underbrace{(A_{sv}(L_{vv}^T)^{-1})^T}_{\text{implicitly known}}$$

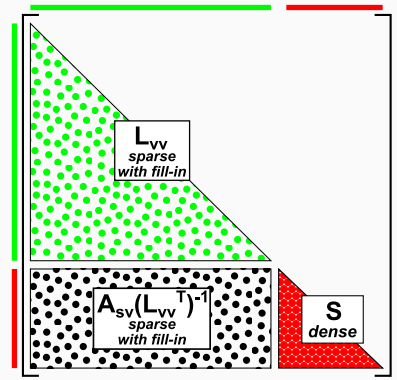


Figure 4: A **AFTER the Schur complement computation**

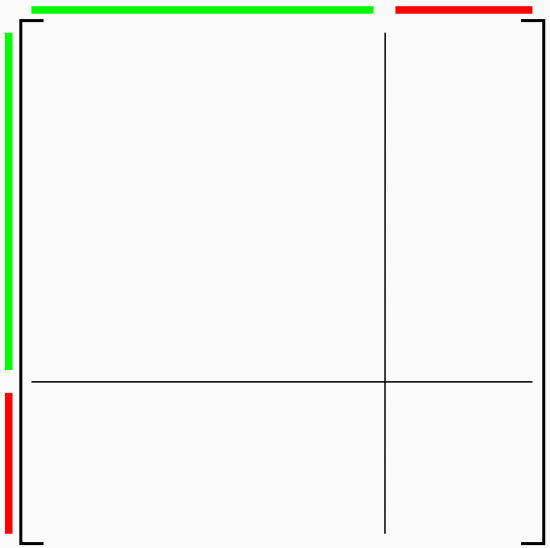
Problem

Without memory constraints

- **two-stage scheme**: coupling of a sparse (MUMPS, PaStiX) and a dense solver (SPIDO, HMAT)
- benefit from community solvers
- Schur complement API $\rightarrow S = A_{ss} - A_{sv} A_{vv}^{-1} A_{sv}^T$

How to proceed with bigger systems?

- do not fit in memory $\rightarrow$ out-of-core
  - missing support in the API
- adapt the **two-stage** scheme
  - ☺ keep using community solvers
  - ☹ loose some advantages

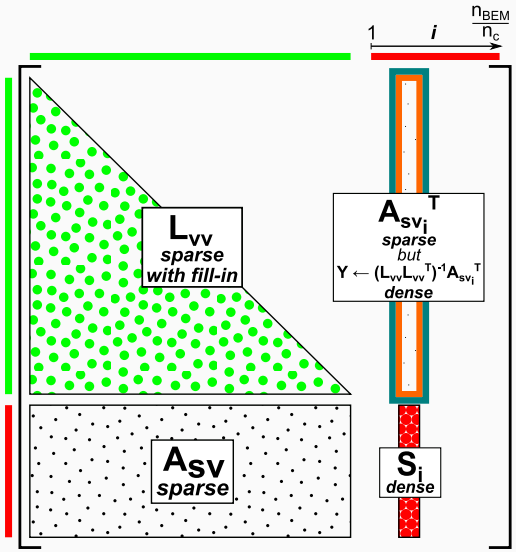


Two-stage implementations

Multi-solve (A)

$$S_i = A_{ss_i} - A_{sv}(L_{vv}L_{vv}^T)^{-1}A_{sv_i}^T$$

- $A_{sv_i}^T$ explicitly stored → broken symmetry
- result of the triangular solve $(L_{vv}L_{vv}^T)^{-1}A_{sv_i}^T$ is a dense matrix

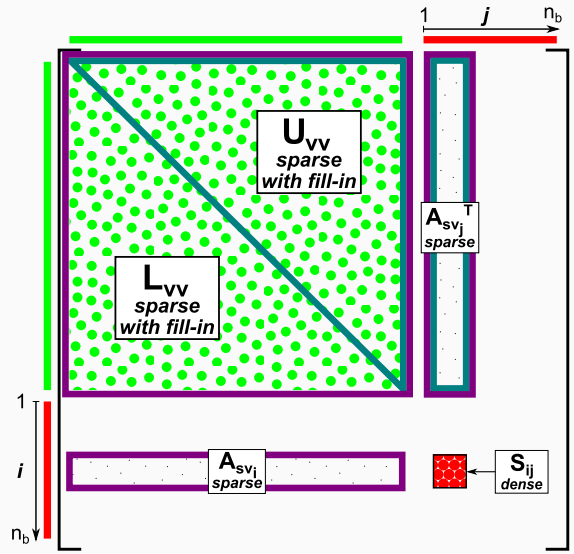


Multi-factorization (B)

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

Schur from API:

- duplicated storage of A_{vv}, A_{sv_i} and $A_{sv_j}^T$ → broken symmetry (except when $i = j$)
- LU instead of LL^T



Preliminary study

Framework

HPC platform

- PlaFRIM, *miriel* nodes
  - 24 cores, 126 GiB of RAM, 293 GiB spinning HDD

Literate [7] and reproducible environment

- clear, exhaustive and accessible documentation
  - `mfelsoci.gitlabpages.inria.fr/thesis/`
- Guix for reproducible software environment

Experimental setup

Test case

- short pipe (length: 2 m, radius: 4 m)
- linear system close enough to real life models
  - reproducible example for the community
- volume mesh v discretized with FEM
- surface mesh s discretized with BEM

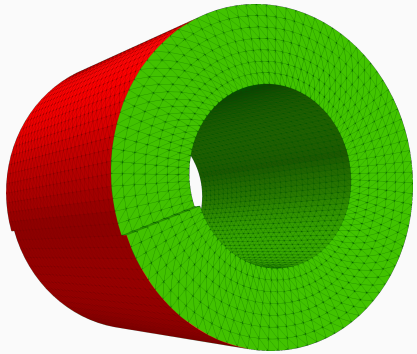


Figure 5: A short pipe test case counting 20,000 unknowns

Configuration

- mainstream HMAT without contributions of A. Falco [6]
- 1 MPI process, 24 OpenMP, MKL threads and StarPU workers
- solution accuracy threshold set to 1×10^{-3} when applicable
- out-of-core disabled

Software	Version	Licensing
GNU Guix [2]	1ac4959c	open-source
HMAT	81db5564	proprietary
HMAT-OSS [3]	4ef1b0ad	open-source
MPF	fec66d43	proprietary
SCAB	297fe52c	proprietary
gcc [1]	9.3.0	open-source
OpenMPI	4.0.3	open-source
Intel(R) MKL	2019.1.144	proprietary
MUMPS [4]	5.2.1	open-source
StarPU [5]	1.3.5	open-source

Results

Best computation times on coupled FEM/BEM systems

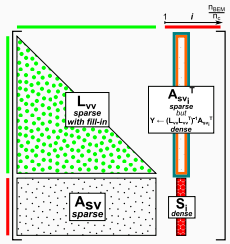


Figure 6: Multi-solve (A)

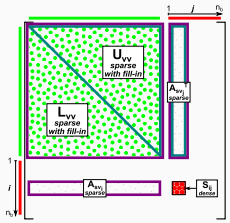
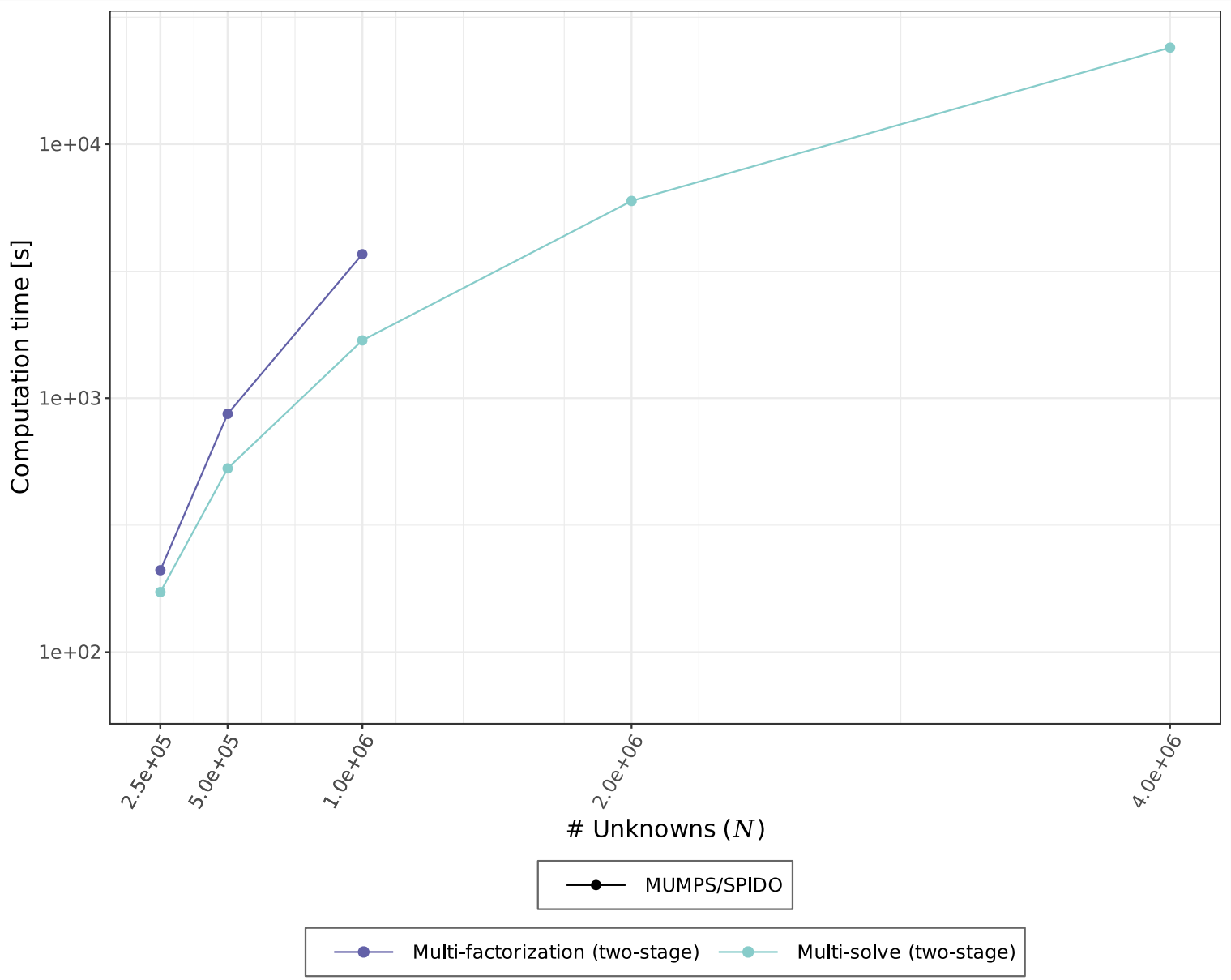


Figure 7: Multi-factorization (B)



Best computation times on coupled FEM/BEM systems

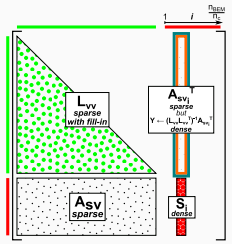


Figure 8: Multi-solve (A)

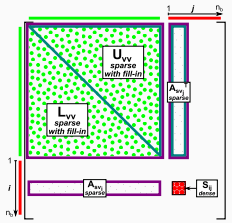
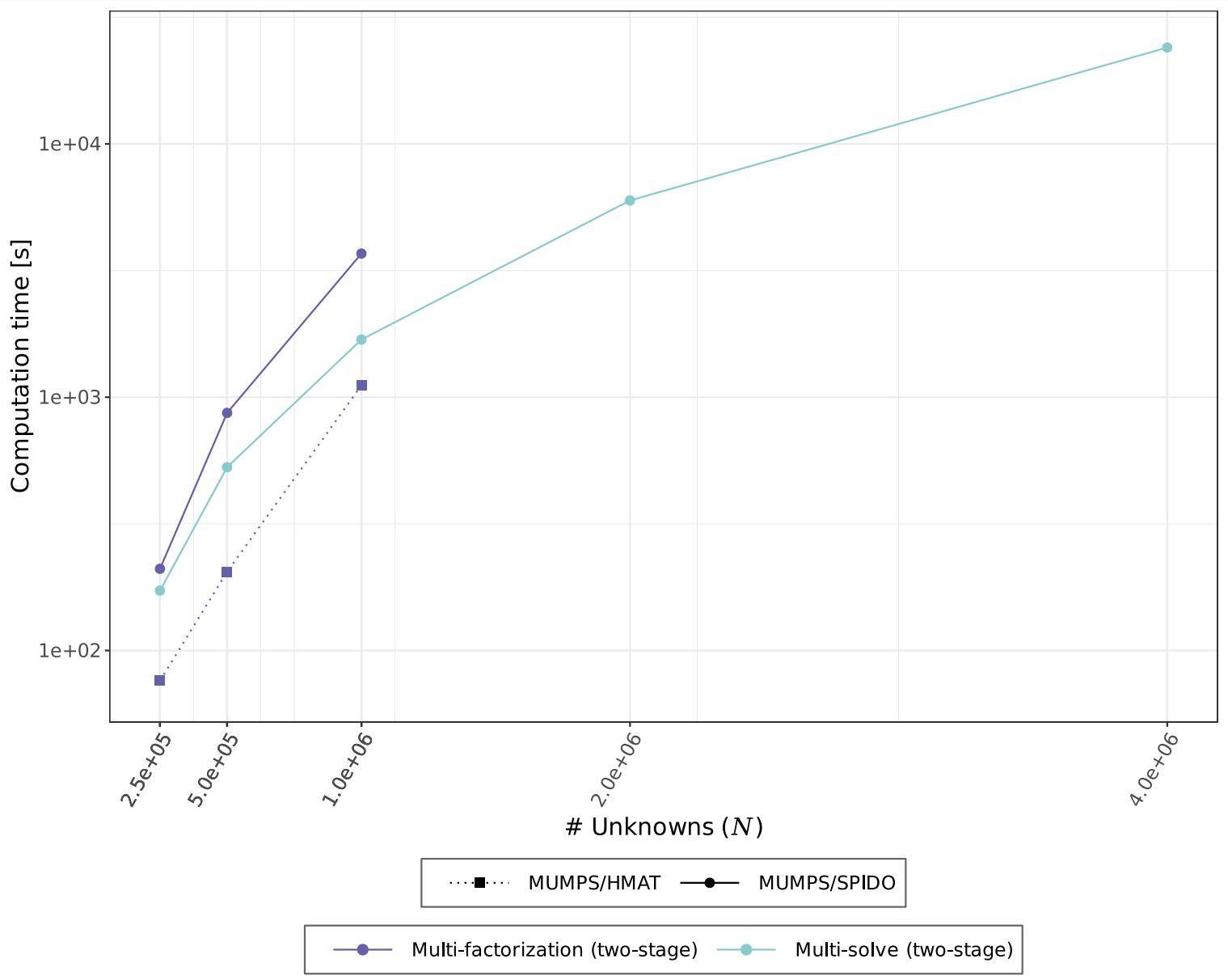


Figure 9: Multi-factorization (B)



Best computation times on coupled FEM/BEM systems

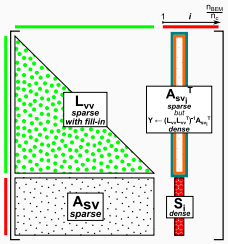


Figure 10: Multi-solve (A)

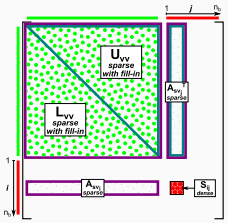
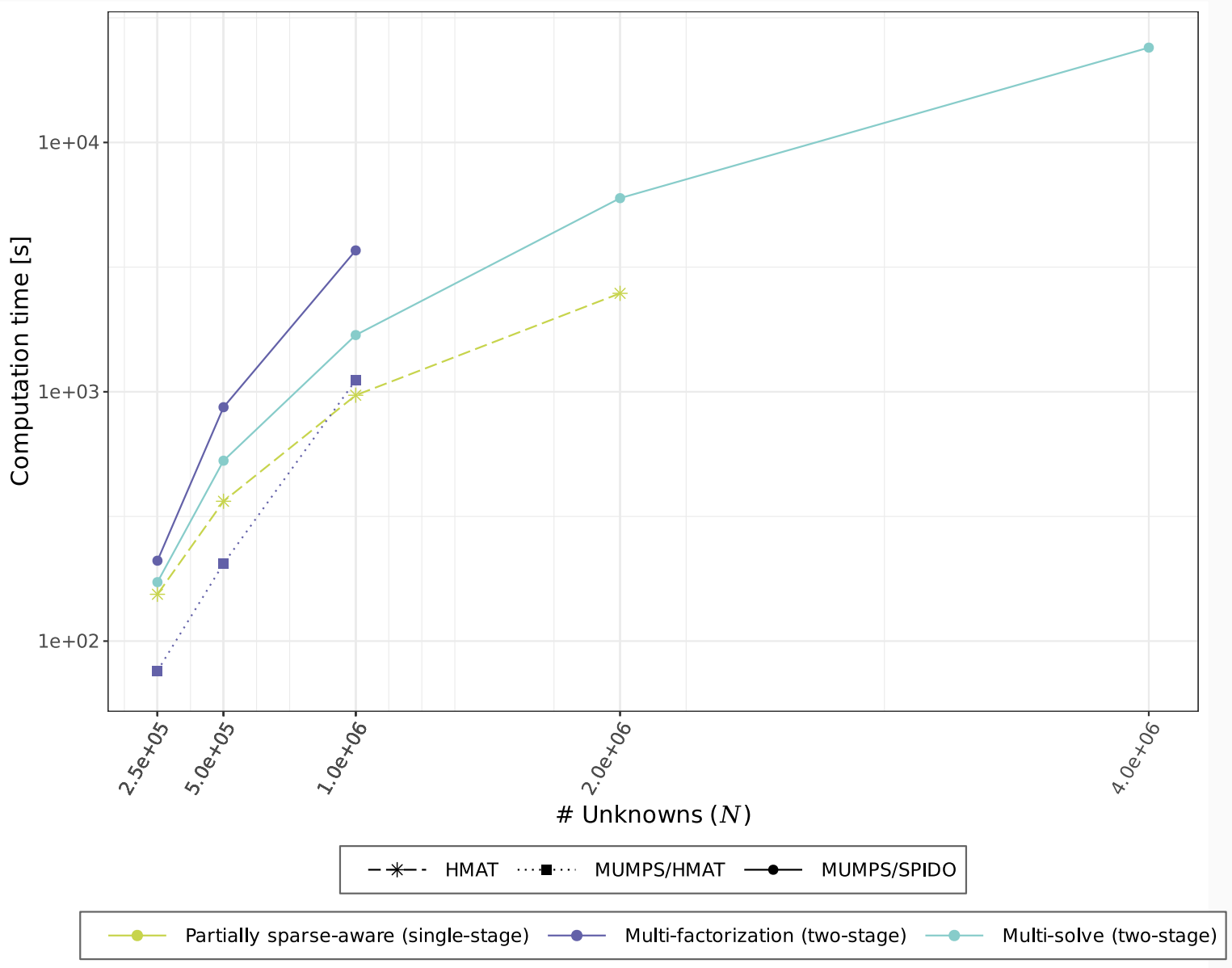


Figure 11: Multi-factorization (B)



Conclusion

Summary (1/2)

Contributions

- a complete preliminary study yielding a research report
  - an extensible basis for the thesis
- automated reproducible and documented benchmark framework

Additional contributions (not discussed here)

- benchmark campaign to evaluate solvers separately on sparse FEM and dense BEM systems

Summary (2/2)

Perspectives

- evaluate the impact of out-of-core
- implant other solver alternatives for the two-stage implementation schemes (e. g. PaStiX and qr_mumps)
- further work on the single-stage schemes in the aim of synthesizing the ideal implementation scheme

Appendix: literate programming

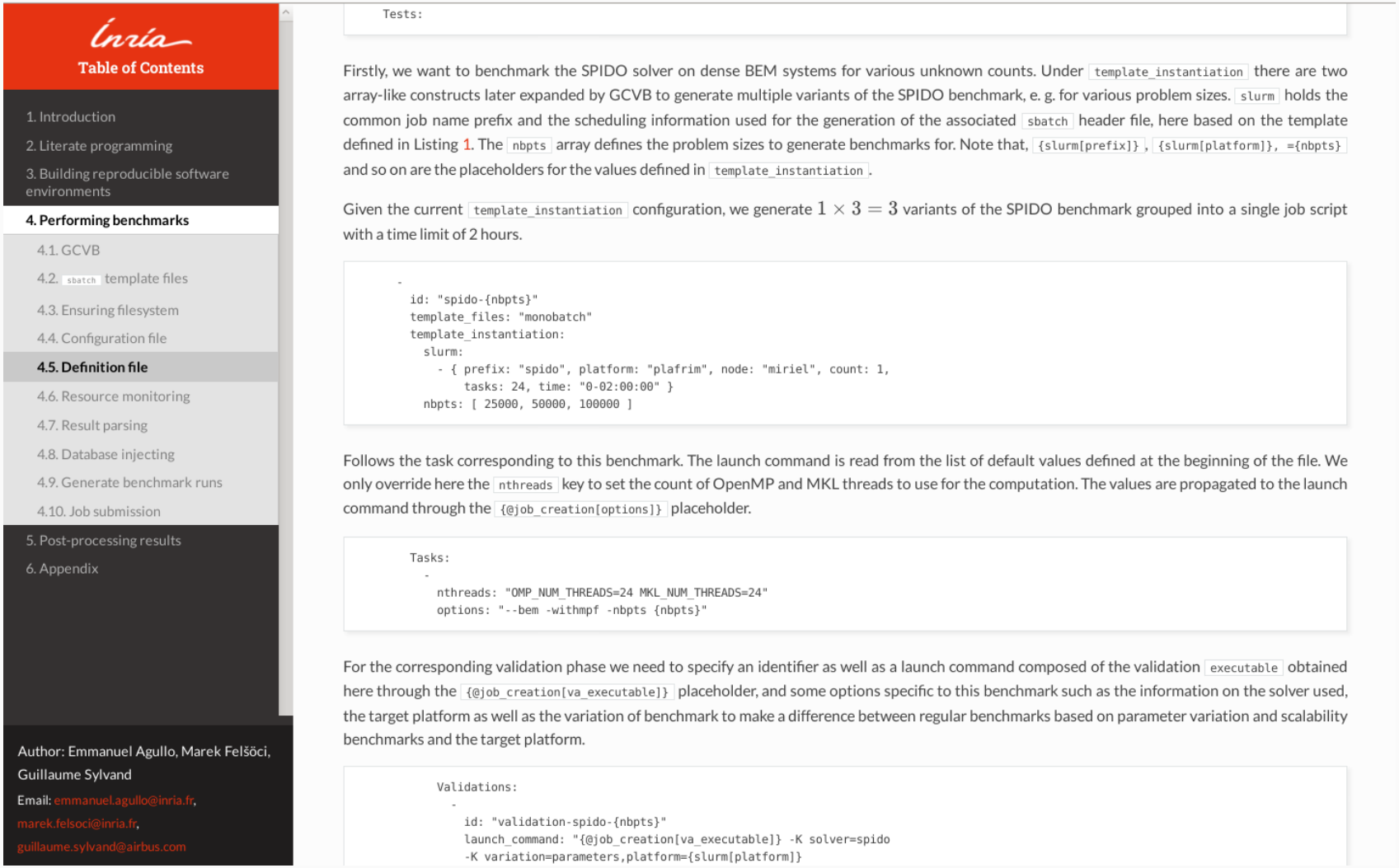


Table of Contents

Author: Emmanuel Agullo, Marek Felšöci, Guillaume Sylvand
Email: emmanuel.agullo@inria.fr, marek.felsoci@inria.fr, guillaume.sylvand@airbus.com


```
Tests:
```

Firstly, we want to benchmark the SPIDO solver on dense BEM systems for various unknown counts. Under `template_instantiation` there are two array-like constructs later expanded by GCVB to generate multiple variants of the SPIDO benchmark, e. g. for various problem sizes. `slurm` holds the common job name prefix and the scheduling information used for the generation of the associated `sbatch` header file, here based on the template defined in Listing 1. The `nbpts` array defines the problem sizes to generate benchmarks for. Note that, `{slurm[prefix]}`, `{slurm[platform]}`, `={nbpts}` and so on are the placeholders for the values defined in `template_instantiation`.

Given the current `template_instantiation` configuration, we generate $1 \times 3 = 3$ variants of the SPIDO benchmark grouped into a single job script with a time limit of 2 hours.

```
-
  id: "spido-{nbpts}"
  template_files: "monobatch"
  template_instantiation:
    slurm:
      - { prefix: "spido", platform: "plafrim", node: "miriel", count: 1,
          tasks: 24, time: "0-02:00:00" }
    nbpts: [ 25000, 50000, 100000 ]
```

Follows the task corresponding to this benchmark. The launch command is read from the list of default values defined at the beginning of the file. We only override here the `nthreads` key to set the count of OpenMP and MKL threads to use for the computation. The values are propagated to the launch command through the `{@job_creation[options]}` placeholder.

```
Tasks:
  -
    nthreads: "OMP_NUM_THREADS=24 MKL_NUM_THREADS=24"
    options: "--bem -withmpf -nbpts {nbpts}"
```

For the corresponding validation phase we need to specify an identifier as well as a launch command composed of the validation `executable` obtained here through the `{@job_creation[va_executable]}` placeholder, and some options specific to this benchmark such as the information on the solver used, the target platform as well as the variation of benchmark to make a difference between regular benchmarks based on parameter variation and scalability benchmarks and the target platform.

```
Validations:
  -
    id: "validation-spido-{nbpts}"
    launch_command: "{@job_creation[va_executable]} -K solver=spido
    -K variation=parameters,platform={slurm[platform]}
```

Figure 12: **Literate description of the experimental software environment (HTML version)**

Acknowledgements

Thank you!

- to the maintainer of Guix, Ludovic Courtès, for his support in deploying our experimental framework
- to Jérôme Robert and Jean-Marie Couteyen from Airbus for their help with the solvers and the benchmark tools developed internally
- to the PlaFRIM team, especially François Rué, for the support with the problems we run into while using the platform

References

[1] *GNU C Compiler*.
`https://gcc.gnu.org/`.

[2] *GNU Guix software distribution and transactional package manager*.
`https://guix.gnu.org`.

[3] *hmat-oss, a hierarchical matrix C/C++ library including a LU solver*.
`https://github.com/jeromerobert/hmat-oss`.

[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] A. Falco, *Combler l'écart entre H-Matrices et méthodes directes creuses pour la résolution de systèmes linéaires de grandes tailles*, thèse de doctorat, Université de Bordeaux, June 2019.

[7] D. E. Knuth, *Literate programming*, Comput. J., 27 (1984), p. 97–111.

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