Solution of larger coupled sparse/dense linear systems in an industrial aeroacoustic context

Emmanuel Agullo, Marek Felšöci, Guillaume Sylvand

June 15, 2022

Inria Bordeaux Sud-Ouest, France
Team-project HiePACS

Introduction

AIRBUS

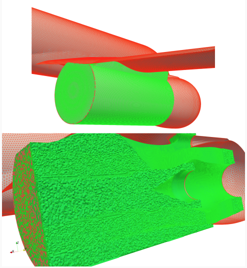
- study the propagation of sound waves emitted by an aircraft
  - acoustic pollution reduction, prototype certification
- discrete model for numerical simulations
  - volume domain v (jet flow)
    - Finite Elements Method (FEM) [19, 16]
  - surface domain s (surface of the aircraft and the volume domain)
    - Boundary Elements Method (BEM) [12, 21]

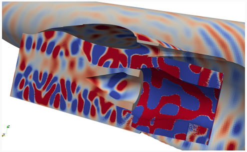
An acoustic wave (blue arrow) emitted by the aircraft's engine, reflected on the wing and crossing the jet flow. Real-life case [20] (left) and a numerical model example (right).

Problem

Global linear system coupling [13, 14] the FEM and the BEM unknowns:

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



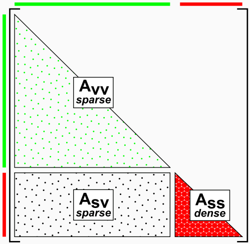


Problem

Global linear system coupling [13, 14] 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
    - lot of zeros → storing only non-zero values
    - discretization of v with FEM (A_{vv}), s/v interaction (A_{sv})
  - a dense part
    - a few or no zeros → storing all values
    - discretization of s with BEM (A_{ss})

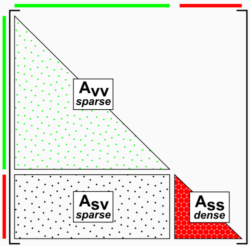


Problem

Global linear system coupling [13, 14] the FEM and the BEM unknowns:

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

- finer model → larger system
- need for efficient solution methods
  - iterative
    - compute a sequence of terms based on previous ones
    - find a good solution approximation
  - direct
    - using Schur complement [22]



Direct solution

Schur complement

- reduce the problem on boundaries → simplify the system to solve

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

Computation steps

1. eliminate x_v from the second equation → Schur complement S

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

2. solve the reduced Schur complement system

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

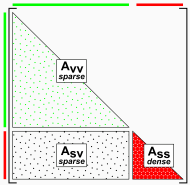
3. determine x_v using x_s

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

Numerical computation

Properties of the input linear system

- A_{vv} and A_{ss} are symmetric
  - storing only half of the coefficients
- A_{vv} and A_{sv} are sparse
  - storing only non-zero values



Initial state of A

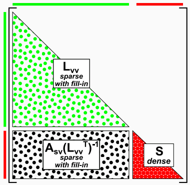
Ideal computation of $S = A_{ss} - A_{sv} A_{vv}^{-1} A_{vs}$

- factorization of A_{vv} into $L_{vv} L_{vv}^T \rightarrow$ 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

Two-stage implementations

Implementation

- coupling of a sparse direct and a dense direct solver
    - fully-featured community solvers with appealing functionalities
        - low-rank compression
        - out-of-core computation
        - 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

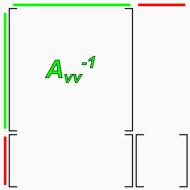
***baseline* coupling**

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

Vanilla solver couplings

***baseline* coupling**

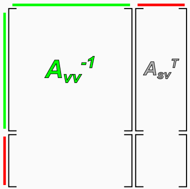
- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



Vanilla solver couplings

***baseline* coupling**

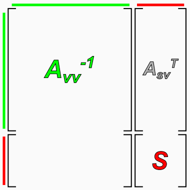
- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



Vanilla solver couplings

***baseline* coupling**

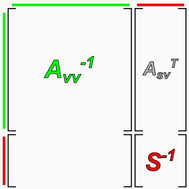
- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



Vanilla solver couplings

***baseline* coupling**

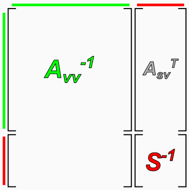
- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



Vanilla solver couplings

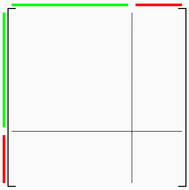
***baseline* coupling**

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



***advanced* coupling**

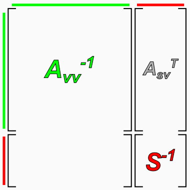
- A as a whole
- *sparse facto.+Schur*
- *dense facto.*, *dense solve*



Vanilla solver couplings

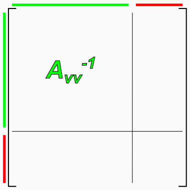
baseline **coupling**

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



advanced **coupling**

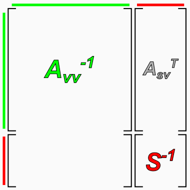
- A as a whole
- *sparse facto.+Schur*
- *dense facto.*, *dense solve*



Vanilla solver couplings

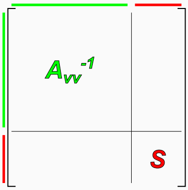
baseline coupling

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



advanced coupling

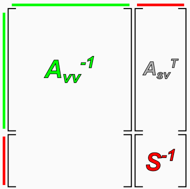
- A as a whole
- *sparse facto.+Schur*
- *dense facto.*, *dense solve*



Vanilla solver couplings

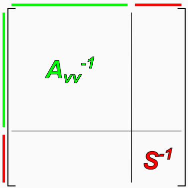
***baseline* coupling**

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



***advanced* coupling**

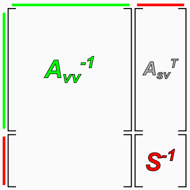
- A as a whole
- *sparse facto.+Schur*
- *dense facto.*, *dense solve*



Vanilla solver couplings

***baseline* coupling**

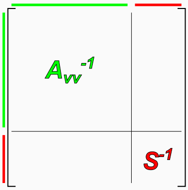
- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



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

***advanced* coupling**

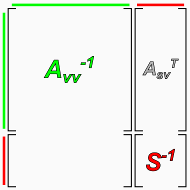
- A as a whole
- *sparse facto.+Schur*
- *dense facto.*, *dense solve*



Vanilla solver couplings

baseline **coupling**

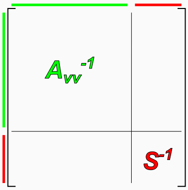
- separate A_{vv}, A_{sv} and A_{ss}
- *sparse facto.*, *sparse solve*
- *dense facto.*, *dense solve*



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

advanced **coupling**

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



- S non-compressed, dense, entirely stored in RAM

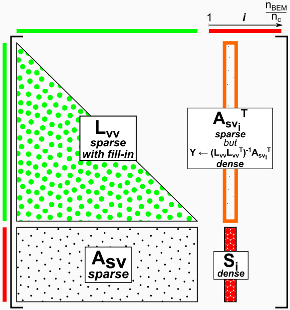
Coping with limitations

- 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 and out-of-core
  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)

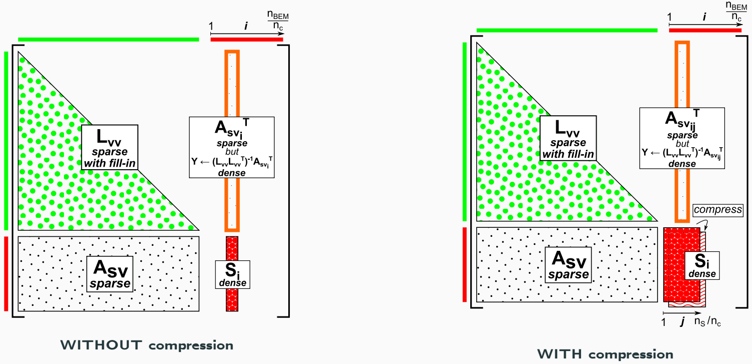


WITHOUT compression

Multi-solve

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

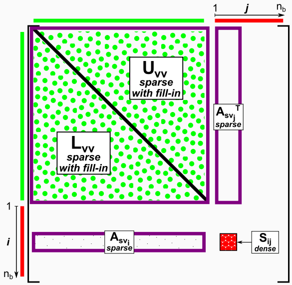
- 1 *sparse facto.* of the green matrix (symmetric)
- plenty of *sparse solve* involving the orange blocks (result is dense)



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

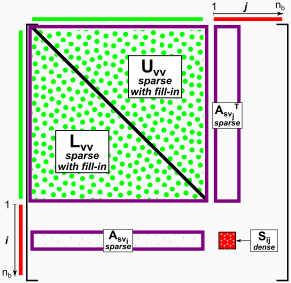


WITHOUT compression

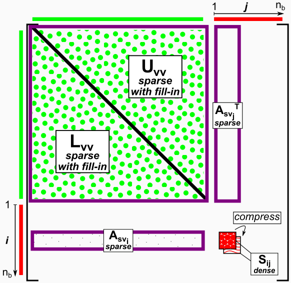
Multi-factorization

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

- 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 [5]

- linear systems close enough to real-life
- arbitrary large FEM/BEM systems

Industrial test case

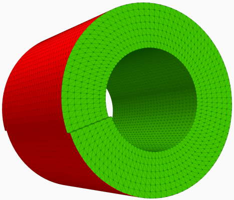
- 2,259,468 unknowns (larger s part)

Solvers

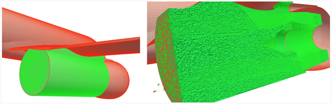
- sparse: MUMPS (compressed) [10]
- dense: SPIDO (non-compressed), HMAT (compressed) [17]

Computation platform

- PlaFRIM [3]



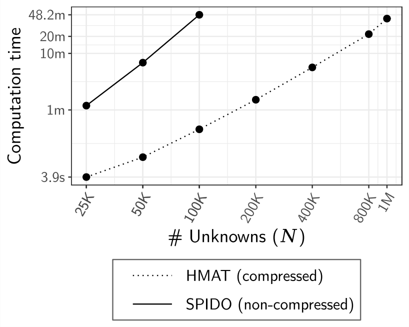
Academic pipe mesh with v and s parts (length: 2 m; radius: 4 m; 20,000 unknowns)

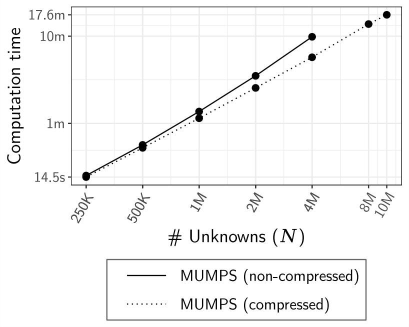


Real-life industrial FEM/BEM mesh

Preliminary comparative study [7]

- single node multi-core benchmarks without out-of-core
- study the solvers separately on sparse FEM and dense BEM systems
  - evaluate the impact of compression
  - identify the best performing parallel configurations
- better understand the behavior on coupled FEM/BEM systems

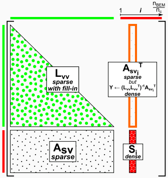




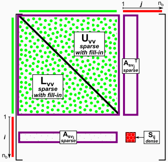
Focus on multi-solve and multi-factorization [8, 9]

- single node multi-core benchmarks without out-of-core
- push the algorithms to their limits (RAM)
- evaluate the impact of compressing the Schur complement S
- study the performance-memory tradeoff for varying block sizes
- validate the algorithms on a real-life industrial case

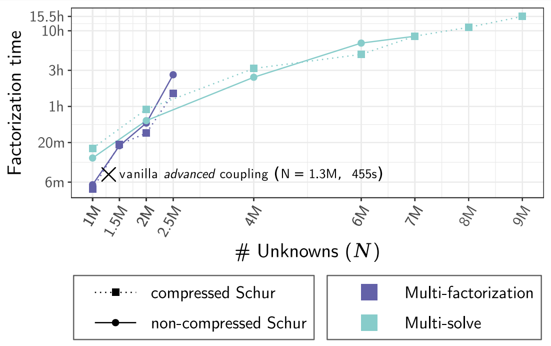
Focus on multi-solve and multi-factorization [8, 9]



multi-solve



multi-factorization



Industrial case

- cannot be processed without our algorithms
- compression of S helps
  - multi-solve: 1.6× faster, 6.4× less RAM
  - multi-factorization: 9.4× faster, 2.0× less RAM

Going further

Energetic profile [6]

- with H. Mathieu (SED), A. Guermouche and B. Tagliaro (STORM)
    - `energy_scope` [18]
- visualize the energy consumption of a complex HPC application
- compare different indicators at once (energy, RAM, flops)
- clues on how to improve the implementation

Out-of-core and distributed memory parallelism (ongoing work)

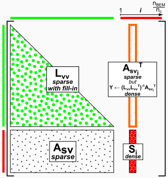
- extends the previous studies of multi-solve and multi-factorization
    1. low-rank compression of S
    2. out-of-core computation of S
    3. scale to multiple computation nodes with MPI

Summary

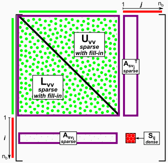
- two algorithms allowing us to:
  - benefit from the most advanced functionalities of fully-featured solvers
  - process larger systems compared to vanilla couplings
    - 9M (multi-solve) and 2.5M (multi-factorization) vs. 1.3M on a single 24-core, 128 GiB RAM workstation
- confirm the advantage of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

Single-stage implementations

Towards ideal implementation



multi-solve

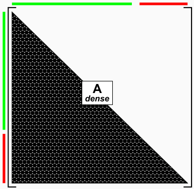


multi-factorization

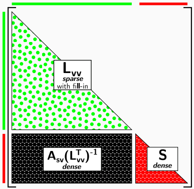
Limitations

- multi-solve: explicit storage of orange blocks in a non-compressed dense matrix
- multi-factorization: superfluous re-factorizations of the sparse submatrix A_{vv}
- two separate stages
  1. Schur complement S assembly
  2. factorization of S and solution of x_s and x_v

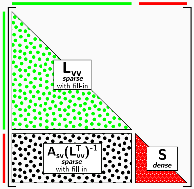
Single-stage schemes



Sparse-oblivious



Partially sparse-aware



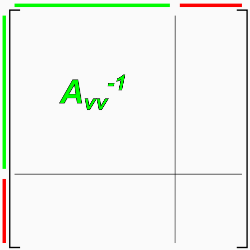
Sparse-aware

Sparse-aware single-stage implementation

- with A. Buttari and A. Jego
    - IRIT/ENSEEIHT, Toulouse
- coupling of task based direct solvers
    - sparse: `qr_mumps` [4]
        - no compression, no distributed memory parallelism (ongoing Ph.D.)
    - dense: HMAT
    - relying on the StarPU runtime [11]
        - 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

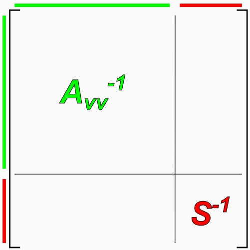
Sparse-aware single-stage implementation

- with A. Buttari and A. Jego
  - IRIT/ENSEEIHT, Toulouse
- coupling of task based direct solvers
  - sparse: `qr_mumps` [4]
    - no compression, no distributed memory parallelism (ongoing Ph.D.)
  - dense: HMAT
  - relying on the StarPU runtime [11]
    - 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



Sparse-aware single-stage implementation

- with A. Buttari and A. Jego
  - IRIT/ENSEEIHT, Toulouse
- coupling of task based direct solvers
  - sparse: `qr_mumps` [4]
    - no compression, no distributed memory parallelism (ongoing Ph.D.)
  - dense: HMAT
  - relying on the StarPU runtime [11]
    - 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



Current status

Done

1. integrate `qr_mumps` into the Airbus solver stack
2. implement multi-solve using `qr_mumps` as sparse solver for matrices with real coefficients
3. implement LL^T factorization for complex symmetric matrices in `qr_mumps` (mission in Toulouse)

Ongoing

4. add a Schur complement API to `qr_mumps`
5. implement multi-factorization using `qr_mumps` as sparse solver

Pending

6. single-stage implementation

Summary

- two-stage multi-solve and multi-factorization allowing us to:
  - benefit from the most advanced functionalities of fully-featured solvers
  - process larger systems compared to vanilla couplings
  - not ideal
- single-stage scheme
  - towards a proof of concept with some sacrifices

Conclusion

Thank you for attending!

References i

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

[2] *Org mode for Emacs*. https://orgmode.org/.

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

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

[5] *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.

References ii

[6] E. Agullo, M. Felšöci, A. Guermouche, H. Mathieu, G. Sylvand, and B. Tagliaro, *Study of the processor and memory power consumption of coupled sparse/dense solvers*, Research Report 9463, Inria Bordeaux Sud-Ouest, Feb. 2022.

[7] E. Agullo, M. Felšöci, and G. Sylvand, *A comparison of selected solvers for coupled FEM/BEM linear systems arising from discretization of aeroacoustic problems*, Research Report RR-9412, Inria Bordeaux Sud-Ouest, June 2021.

[8] ——, *Comparison of coupled solvers for FEM/BEM linear systems arising from discretization of aeroacoustic problems*, in COMPAS 2021 - Conférence francophone d'informatique en Parallélisme, Architecture et Système, Lyon / Virtuel, France, July 2021.

References iii

[9] ——, *Direct solution of larger coupled sparse/dense linear systems using low-rank compression on single-node multi-core machines in an industrial context*, Research Report 9453 (accepted at IPDPS 2022), Inria Bordeaux Sud-Ouest, Feb. 2022.

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

[11] 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.

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

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

[14] 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.

[15] C. Dominik, *The Org Mode 9.1 Reference Manual*, 12th Media Services, 2018.

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

[17] 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.

[18] H. Mathieu, *Energy Scope: a tool for measuring the energy profile of HPC and AI applications*. https://jcad2021.sciencesconf.org/data/Herve_Mathieu_energy_scope.pdf, **2021.**

[19] 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.

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

[21] 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.

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

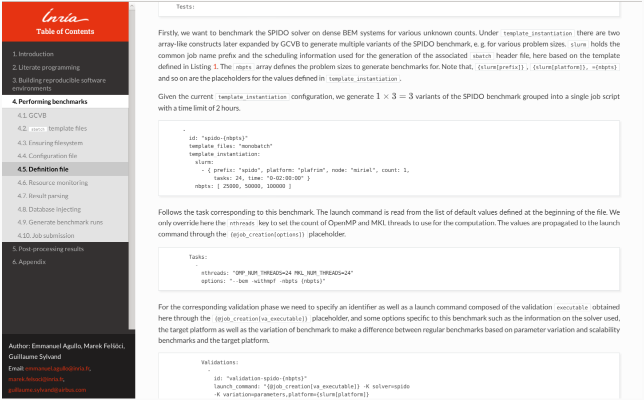
Appendix

Reproducible software envionments with Guix [1]

Guix

- transactional package manager able to co-exist with the primary package manager
- self-contained, executable descriptions of entire software environments
- reproducible accross multiple different machines
    - natively or through container solutions

Reproducible studies with Org mode for Emacs [2, 15]



Inria
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]}
```

- literate programming paradigm
  - combining formatted text with source code
- exhaustive documentation allowing others to reproduce a study
  - question of proprietary source code, e.g. Airbus

Going further

How to build a reproducible study from scratch with Guix and Org

- `tuto-techno-guix-hpc.gitlabpages.inria.fr/guidelines/`