Algebraic Multigrid Methods for Simulation of Single Phase Flow in Fractured Media

MS13 - Advances in Block Preconditioners and Multilevel Methods for Numerical PDEs
SIAM Conference on Applied Linear Algebra (LA24)

Fabio Durastante (fabio.durastante@unipi.it)

University of Pisa,

Department of Mathematics.

Consiglio Nazionale delle Ricerche,

Istituto per le Applicazioni del Calcolo "M. Picone."

Paris — May 13, 2024

Dipartimento di Matematica
Università di Pisa

Collaborators & Funding

1 With a Little Help from My Friends

Pasqua D'Ambra,
Consiglio Nazionale delle Ricerche
Istituto per le Applicazioni del Calcolo
"M. Picone"

Salvatore Filippone,
Università degli Studi di Roma "Tor Vergata"
Dipartimento di Ingegneria Civile e
Ingegneria Informatica
IAC-CNR

"INdAM - GNCS Project" CUP E53C23001670001.

HORIZON-EUROPEHPC-JU-2023-COE-01
Agreement N.101144014

This work has been supported by the **Spoke 1** "FutureHPC & BigData" of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR Missione 4 Componente 2 Investimento 1.4: Potenziamento strutture di ricerca e creazione di "campioni nazionali di R&S (M4C2-19)", and the **Spoke 6** "Multiscale modelling & Engineering applications" - Next Generation EU (NGEU).

Table of Contents

The Single Phase Problem

Highly fractured andesite, with an altered vein. Yalour Islands, Wilhelm Archipelago, Antarctica – Luis Bartolomé Marcos.

Our problem:

- An **incompressible flow**,
- in a **mixed-dimensional geometry**,
- with a Darcy-type relation between the flux and the pressure gradient.

Highly fractured andesite, with an altered vein. Yalour Islands, Wilhelm Archipelago, Antarctica – Luis Bartolomé Marcos.

Our problem:

- An **incompressible flow**,
- in a **mixed-dimensional geometry**,
- with a Darcy-type relation between the flux and the pressure gradient.

The **set of equations** is:

$$\begin{aligned} \operatorname{div} \mathbf{u} &= s, && \text{in } \Omega, \\ \mathbf{u} &= -K \nabla p, && \text{in } \Omega, \\ p &= \overline{p}, && \text{on } \Gamma, \end{aligned}$$

Formulating the problem

$$\begin{aligned} \operatorname{div} \mathbf{u} &= s, && \text{in } \Omega, \\ \mathbf{u} &= -K\nabla p, && \text{in } \Omega, \\ p &= \overline{p}, && \text{on } \Gamma, \end{aligned}$$

- $\mathbf{u}$ fluid flux,

Formulating the problem

$$
\begin{aligned}
\operatorname{div} \mathbf{u} &= s, && \text{in } \Omega, \\
\mathbf{u} &= -K \nabla p, && \text{in } \Omega, \\
p &= \overline{p}, && \text{on } \Gamma,
\end{aligned}
$$

- $\mathbf{u}$ fluid flux,
- p pressure,

Formulating the problem

$$
\begin{aligned}
\operatorname{div} \mathbf{u} &= s, && \text{in } \Omega, \\
\mathbf{u} &= -K \nabla p, && \text{in } \Omega, \\
p &= \overline{p}, && \text{on } \Gamma,
\end{aligned}
$$

- $\mathbf{u}$ fluid flux,
- p pressure,
- s source term,

Formulating the problem

$$
\begin{aligned}
\operatorname{div} \mathbf{u} &= s, && \text{in } \Omega, \\
\mathbf{u} &= -K \nabla p, && \text{in } \Omega, \\
p &= \overline{p}, && \text{on } \Gamma,
\end{aligned}
$$

- $\mathbf{u}$ fluid flux,
- p pressure,
- s source term,
- K hydraulic conductivity tensor,

Formulating the problem

$$
\begin{aligned}
\operatorname{div} \mathbf{u} &= s, && \text{in } \Omega, \\
\mathbf{u} &= -K\nabla p, && \text{in } \Omega, \\
p &= \overline{p}, && \text{on } \Gamma,
\end{aligned}
$$

- $\mathbf{u}$ fluid flux,
- p pressure,
- s source term,
- K hydraulic conductivity tensor,
- $\overline{p}$ pressure on the boundary.

Formulating the problem

$$\begin{aligned} \operatorname{div} \mathbf{u} &= s, && \text{in } \Omega, \\ \mathbf{u} &= -K\nabla p, && \text{in } \Omega, \\ p &= \overline{p}, && \text{on } \Gamma, \end{aligned}$$

- $\mathbf{u}$ fluid flux,
- p pressure,
- s source term,
- K hydraulic conductivity tensor,
- $\overline{p}$ pressure on the boundary.

We add a **fracture network** dividing Ω in a set of subdomains Ω_i, i.e., $\Omega = \cup_i \Omega_i$.

- If $\Omega \subseteq \mathbb{R}^2$ then the Ω_i are divided by **interfaces** of dimensions $\{0, 1\}$,
- If $\Omega \subseteq \mathbb{R}^3$ then the Ω_i are divided by **interfaces** of dimensions $\{0, 1, 2\}$.

Formulating the problem

$$
\begin{aligned}
\operatorname{div} \mathbf{u} &= s, && \text{in } \Omega, \\
\mathbf{u} &= -K\nabla p, && \text{in } \Omega, \\
p &= \overline{p}, && \text{on } \Gamma,
\end{aligned}
$$

- $\mathbf{u}$ fluid flux,
- p pressure,
- s source term,
- K hydraulic conductivity tensor,
- $\overline{p}$ pressure on the boundary.

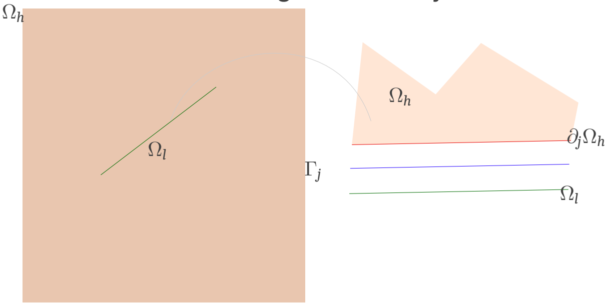
We add a **fracture network** dividing Ω in a set of subdomains Ω_i, i.e., $\Omega = \cup_i \Omega_i$.

- If $\Omega \subseteq \mathbb{R}^2$ then the Ω_i are divided by **interfaces** of dimensions $\{0, 1\}$,
- If $\Omega \subseteq \mathbb{R}^3$ then the Ω_i are divided by **interfaces** of dimensions $\{0, 1, 2\}$.

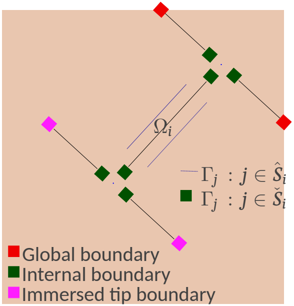
we need to "break" the equation across all the subdomains.

A fractured domain

Mixed-dimensional geometric objects



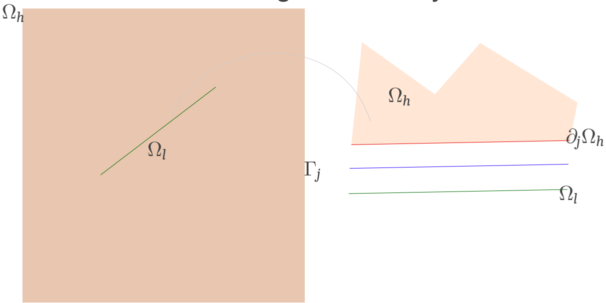
Entities of the mixed-dimensional geometry.



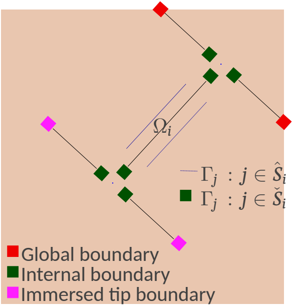
- Ω_h and Ω_l two subdomains one dimension apart,
- $\partial_j\Omega_h$ be the part of the boundary of $\partial\Omega_h$ that coincides with Ω_l,
- Ω_j interface on the boundary between $\partial\Omega_h$ and Ω_l.

A fractured domain

Mixed-dimensional geometric objects



Entities of the mixed-dimensional geometry.



- Ω_h and Ω_l two subdomains one dimension apart,
- $\partial_j\Omega_h$ be the part of the boundary of $\partial\Omega_h$ that coincides with Ω_l,
- Ω_j interface on the boundary between $\partial\Omega_h$ and Ω_l.
- Γ_j, $\partial_j\Omega_h$ and Ω_l are all geometrically coincident

A fractured set of equations

Assuming $\Omega = \sum_i \Omega_i$ we rewrite the equations as:

$$\mathbf{u}_i + \frac{\mathcal{K}_i}{\mu_i} \nabla p_i = 0$$

$$\operatorname{div} \mathbf{u}_i - \sum_{j \in S} \Xi^i_j \lambda_j = s_i,$$

$$\mathbf{u}_i \cdot \mathbf{n}_i \big|_{\partial j \Omega_j} = \Xi^i_j \lambda_j \qquad \forall j \in \check{S}_i.$$

$$\lambda_j + \frac{\kappa_j}{\mu_j} \left(\Pi^l_j p_l - \Pi^h_j \operatorname{tr} p_h \right) = 0,$$

$$+\text{B.C.}$$

A fractured set of equations

Assuming $\Omega = \sum_i \Omega_i$ we rewrite the equations as:

$$\mathbf{u}_i + \frac{\mathcal{K}_i}{\mu_i} \nabla p_i = 0$$

$$\operatorname{div} \mathbf{u}_i - \sum_{j \in S} \Xi_j^i \lambda_j = s_i,$$

$$\mathbf{u}_i \cdot \mathbf{n}_i |_{\partial j \Omega_j} = \Xi_j^i \lambda_j \qquad \forall j \in \check{S}_i.$$

$$\lambda_j + \frac{\kappa_j}{\mu_j} \left(\Pi_j^l p_l - \Pi_j^h \operatorname{tr} p_h \right) = 0,$$

$+$B.C.

Physical constants:

- $\mathcal{K}_i$ effective tangential permeability tensor scaled by the aperture (Berre et al. 2021),
- μ_i the fluid viscosity,
- κ_i the normal effective permeability.

A fractured set of equations

Assuming $\Omega = \sum_i \Omega_i$ we rewrite the equations as:

$$\mathbf{u}_i + \frac{\mathcal{K}_i}{\mu_i} \nabla p_i = 0$$

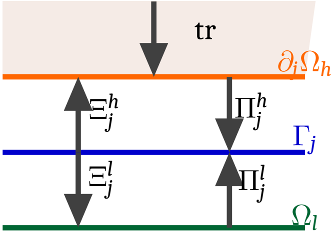
$$\operatorname{div} \mathbf{u}_i - \sum_{j \in S} \Xi_j^i \lambda_j = s_i,$$

$$\mathbf{u}_i \cdot \mathbf{n}_i|_{\partial_j \Omega_j} = \Xi_j^i \lambda_j \qquad \forall j \in \check{S}_i.$$

$$\lambda_j + \frac{\kappa_j}{\mu_j} \left(\Pi_j^l p_l - \Pi_j^h \operatorname{tr} p_h \right) = 0,$$

$$+\text{B.C.}$$

Technicalia to interface domains:



- Flux on Γ_j: $\lambda_j = \Pi_j^h \operatorname{tr} \mathbf{q}_h \cdot \mathbf{n}_h$ for $\mathbf{n}_h$ the unit normal on $\partial_j \Omega_h$ pointing from Ω_h to $\partial_j \Omega_h$

Assembling the matrix

- The **computational grid** is built in such a way to **conform** to all fractures, their intersection lines and points,
- **Mortar FEM** approach with a **second pass** to enforce conformal setting.

</> We use the `PorePy` (Keilegavlen et al. 2021) (`v.1.8.1`) to generate the matrices.

- The **computational grid** is built in such a way to **conform** to all fractures, their intersection lines and points,
- **Mortar FEM** approach with a **second pass** to enforce conformal setting.

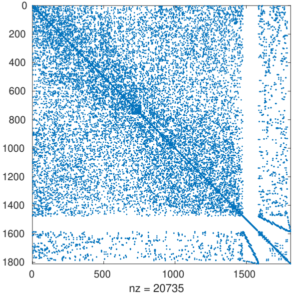
</> We use the `PorePy` (Keilegavlen et al. 2021) (`v.1.8.1`) to generate the matrices.

Example with two subdomains

For **two subdomains** the matrix has the form:

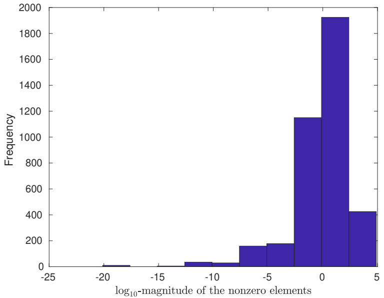
$$\mathcal{A}\mathbf{y} \equiv \begin{bmatrix} A_h & & N_h \Xi_j^h \\ & A_l & S_l \Xi_j^l \\ -\Pi_j^h P_h & \Pi_j^l P_l & M_j \end{bmatrix} \begin{bmatrix} \mathbf{y}_h \\ \mathbf{y}_l \\ \xi_j \end{bmatrix} = \begin{bmatrix} \mathbf{s}_h \\ \mathbf{s}_l \\ \mathbf{0} \end{bmatrix} \equiv \mathbf{s},$$

Assembling the matrix



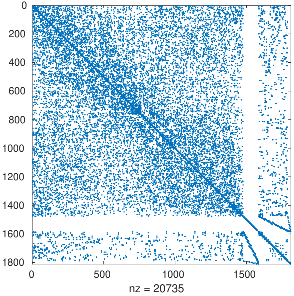
- $\mathcal{A}$ is non symmetric *but* pattern symmetric,
- elements have wide variation in magnitude:
  - both for the resolution of the grid near the fractures,
  - and for the use of realistic physical constants.
- There are some 0s on the diagonals.

Assembling the matrix



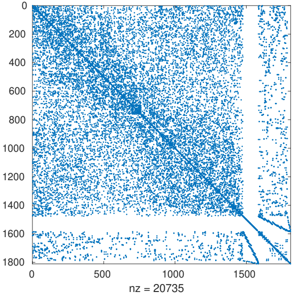
- $\mathcal{A}$ is non symmetric *but* pattern symmetric,
- elements have wide variation in magnitude:
  - both for the resolution of the grid near the fractures,
  - and for the use of realistic physical constants.
- There are some 0s on the diagonals.

Assembling the matrix



- $\mathcal{A}$ is non symmetric *but* pattern symmetric,
- elements have wide variation in magnitude:
  - both for the resolution of the grid near the fractures,
  - and for the use of realistic physical constants.
- There are some 0s on the diagonals.
- We **do not want to interact with the discretization code** to have ordering of the unknowns and division into blocks as in the theory.

Assembling the matrix



- $\mathcal{A}$ is non symmetric *but* pattern symmetric,
- elements have wide variation in magnitude:
  - both for the resolution of the grid near the fractures,
  - and for the use of realistic physical constants.
- There are some 0s on the diagonals.
- We **do not want to interact with the discretization code** to have ordering of the unknowns and division into blocks as in the theory.

Objective

Obtain an **algebraic** and **monolithic solver** for the problem by coupling a Krylov method (GMRES) and an Algebraic Multigrid Preconditioner.

Table of Contents

3 The PSCToolkit Library

Two central libraries PSBLAS and AMG4PSBLAS:

- Existing software standards:
  - MPI, OpenMP, CUDA
  - Serial sparse BLAS,
  - (Par)Metis,
  - AMD
- Attention to performance using modern Fortran;
- Research on new preconditioners;
- No need to delve in the data structures for the user;
- Tools for error and mesh handling beyond simple algebraic operations;
- Standard Krylov solvers

Two central libraries PSBLAS and AMG4PSBLAS:

- Domain decomposition preconditioners
- Algebraic multigrid with aggregation schemes
  - Parallel coupled weighted **matching based aggregation**[1,2]
  - Parallel decoupled smoothed aggregation (**V**aněk, **B**rezina, **M**andel)
- Parallel Smoothers (Block-Jacobi, DD-Schwartz, Hybrid-GS/SGS/FBGS, ℓ_1 variants) that can be coupled with specialized block (approximate) solvers MUMPS, SuperLU, incomplete factorizations (AINV, INVK/L, ILU-type), Polynomial Accelerators (*@Pasqua's Talk - MS103 Thursday 11:50*)
- V-Cycle, W-Cycle, K-Cycle, Variable V-Cycle

1 P. D'Ambra, S. Filippone and P. S. Vassilevski, BootCMatch: a software package for bootstrap AMG based on graph weighted matching, ACM Trans. Math. Software **44** (2018), no. 4, Art. 39, 25 pp.

2 P. D'Ambra, F. D. and S. Filippone, AMG preconditioners for linear solvers towards extreme scale, SIAM J. Sci. Comput. **43** (2021), no. 5, S679–S703.

Two central libraries PSBLAS and AMG4PSBLAS.

- Freely available from: https://psctoolkit.github.io,
- Open Source with BSD 3 Clause License.
- P. D'Ambra, F. D., and S. Filippone, Parallel Sparse Computation Toolkit, Software Impacts (2023): 100463.

```
git clone --recurse-submodules \
git@github.com:psctoolkit/psctoolkit.git
(cd psblas3; ./configure; make -j; make install)
(cd psblas3-ext; ./configure; make -j; make install)
(cd amg4psblas; ./configure; make -j; make install)
```

Given Matrix $A \in \mathbb{R}^{n \times n}$

Wanted Iterative method B to precondition a Krylov method method:

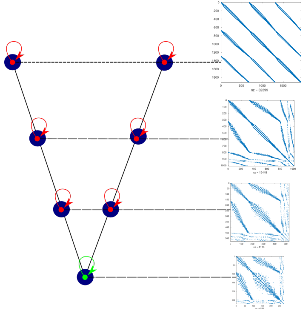
- Hierarchy of systems

$$A_l \mathbf{x}_{=} \mathbf{b}_l, l = 0, \dots, n_{\text{lev}}$$

- Transfer operators:

$$P^l_{l+1} : \mathbb{R}^{n_{l+1}} \rightarrow \mathbb{R}^{n_l}$$

Missing Structural/geometric infos



Smoother: "High frequencies"

$$M_l : \mathbb{R}^{n_l} \rightarrow \mathbb{R}^{n_l}$$

Prolongator: "Low frequencies"

$$P^l_{l+1} : \mathbb{R}^{n_l} \rightarrow \mathbb{R}^{n_{l+1}}$$

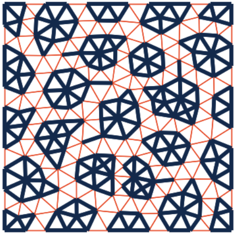
Complementarity of Smoother and Prolongator

Decoupled Vaněk Mandel and Brezina Aggregation

- *Algebraic strength*: nodes with **strong connections** (high coupling) are grouped together.
- Aggregation across different processes is done in a decoupled manner, i.e., each process finds aggregates only in its diagonal block.
- We generate a *disjoint covering* $\{C_i^l\}_{i=1}^{n_{l+1}}$ of the set $\{1,\dots,n_l\}$ that are the aggregates

$$(\hat{P}_l)_{i,j} = \begin{cases} 1, & \text{if } i \in C_j^l, \\ 0, & \text{otherwise.} \end{cases}$$

$$N_i^l(\theta) = \{j \,:\, |a_{i,j}| \geq \theta\sqrt{a_{i,i}a_{j,j}}\},$$
$$\theta \in (0,1), \quad l = 0,\dots,N_\ell - 1.$$

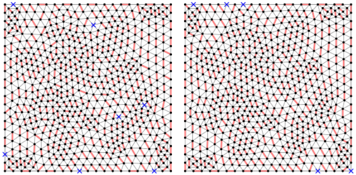


Vaněk, P., J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. Computing 56.3 (1996): 179-196..

D'Ambra, P., D. Di Serafino, and S. Filippone. MLD2P4: a package of parallel algebraic multilevel domain decomposition preconditioners in Fortran 95. TOMS 37.3 (2010): 7-23.

Aggregation via Coupled Parallel Matching Algorithms

1. We **construct the aggregates** by applying maximum weight matching to the graph constructed starting from A.
2. We perform an **approximate global matching** over the whole graph for better aggregation quality.



Algorithm: Locally Dominant Edge

Input: Graph $G = (\mathcal{V}, \mathcal{E})$, Weights $\hat{A}$

$\mathcal{M} \leftarrow \emptyset$;

while $\mathcal{E} \neq \emptyset$ **do**

Take a locally dominant edge $(i, j) \in \mathcal{E}$, i.e., such that

$$\arg\max_k \hat{a}_{ik} = \arg\max_k \hat{a}_{jk} = \hat{a}_{ij}$$

Add $(i, j) \in \mathcal{M}$;

Remove all edges incident to i and j from $\mathcal{E}$;

end

Output: Matching $\mathcal{M}$

- Ü. V. Çatalyürek, F. Dobrian, A. Gebremedhin, M. Halappanavar and A. Pothen, Distributed-Memory Parallel Algorithms for Matching and Coloring, 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum, Anchorage, AK, USA, 2011, pp. 1971-1980, doi: 10.1109/IPDPS.2011.360.
- D'ambra, P., S. Filippone, and P. S. Vassilevski, BootCMatch: a software package for bootstrap AMG based on graph weighted matching. ACM TOMS 44.4 (2018): 1-25.

Change the aggregation to work with diagonal 0s.

To obtain a **more robust aggregation** it is necessary to apply a **smoothing operation** to the interpolation operator, i.e., increasing its regularity *from piecewise constant to something more*:

$$P_l = (I - \omega D_l^{-1} A_l)\hat{P}_l,\ D_l = \mathrm{diag}(A_l) \qquad \text{(Jacobi smoothing)}$$

A_l has 0s on the diagonal $\Rightarrow$ D_l is **not invertible**, hence we use:

$$\left(D_l^{(\ell_1)}\right)_{i,i} = (D_l)_{i,i} + \sum_{j=1 \wedge j \neq i}^{n_l} |(A_l)_{i,j}|.$$

We use $D_l^{(\ell_1)}$ also in the filtering procedure to reduce the number of non-zero elements in the *coarse matrices*.

Hu, J.J., Siefert, C.M., Tuminaro, R.S.: Smoothed aggregation for difficult stretched mesh and coefficient variation problems. Numer. Linear Algebra Appl.29(6),2442–26 (2022).s

Few preliminary tests

Here I present some **preliminary tests** that focus on:

- Algorithmic behavior of the ℓ_1-smoothed aggregation routines,
- Node-level code optimization with OpenMP for the build phase of the preconditioner,
- Solution performance on NVIDIA GPUs.

Tested on the Toeplitz machine (Green Data Center @ UNIPI)

1 `cl1` Node:
Intel® Xeon® CPU E5-2643 v4 at 3.40 GHz with 2 threads per core, 6 core per socket and 2 sockets. The node is equipped with 128 Gb of RAM.

4 `cl2` Nodes:
Intel® Xeon® CPU E5-2650 v4 at 2.20GHz with 2 threads per core, 12 cores per socket and 2 socket. The nodes are equipped with 256 Gb of RAM.

4 `gpu` Nodes:
AMD EPYC 7763 64-Core Processor at 3.5GHz with 2 threads per core, 64 cores per socket and 2 sockets. The nodes are equipped with 2Tb of RAM and 4 NVIDIA A40 GPUs.

The Preconditioners, Solver & Software Environment

VBM A single sweep of a V-cycle AMG with the smoothed aggregation based on the scheme of (Vaněk, Mandel, and Brezina 1996), 4 iterations of ℓ_1-Jacobi method as smoother, and 30 iteration of ℓ_1-Jacobi as coarse solver.

MATCH A single sweep of a V-cycle AMG with the smoothed aggregation based on compatible relaxation based on graph matching and aggregates of size at most 8, 4 iterations of ℓ_1-Jacobi method as smoother, and 30 iteration of ℓ_1-Jacobi as coarse solver.

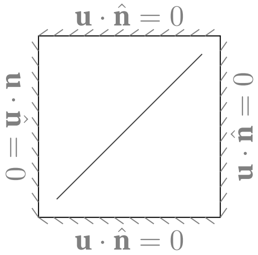
The **solver** is GMRES(30) with a tolerance on the relative residual of $\tau = 10^{-6}$.

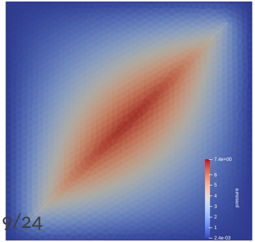
Software environment

On the `cl1` and `cl2` nodes: `gcc/13.2.0`, `metis/5.1.0`, `openblas/0.3.26`, `openmpi/4.1.6`. On the `gpu` nodes `gcc/12.2.0`, `metis/5.1.0`, `openmpi/4.1.6`, `openblas/0.3.26`, `cuda/12.3.1`.

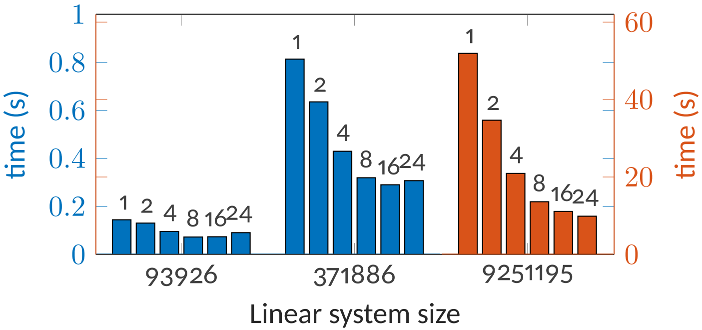
OpenMP Acceleration of the Build Phase

4 Few preliminary tests



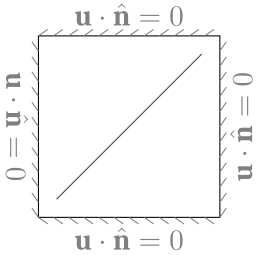


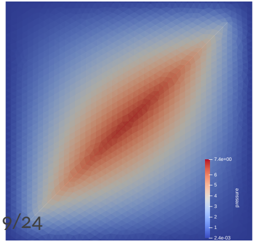
2D test with a single fracture acting as a unit source.



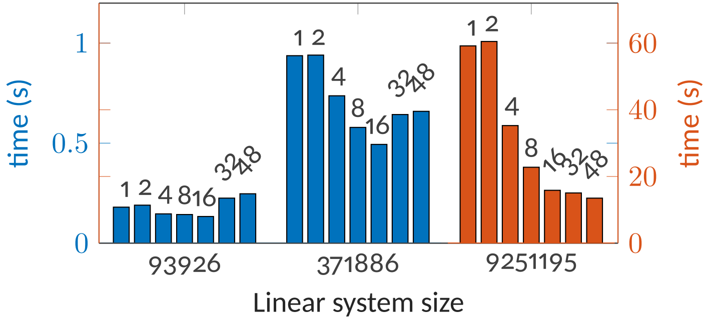
`cl1` node

OpenMP Acceleration of the Build Phase

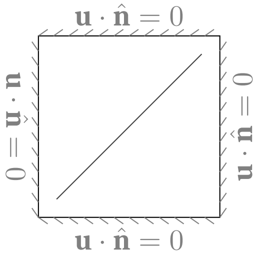


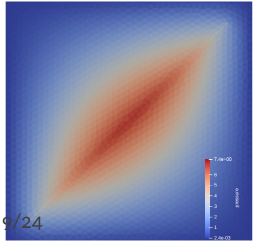


2D test with a single fracture acting as a unit source.

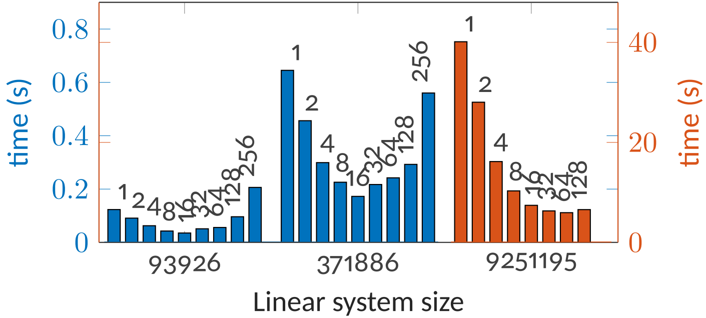


cl2 node



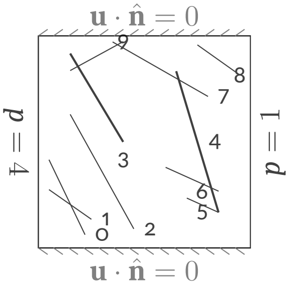


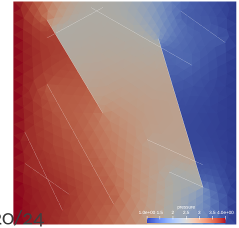
2D test with a single fracture acting as a unit source.



gpu node

Algorithmic Scalability (Pure MPI)





2D case, several fractures high-difference in permeability.

Iteration count.

VBM

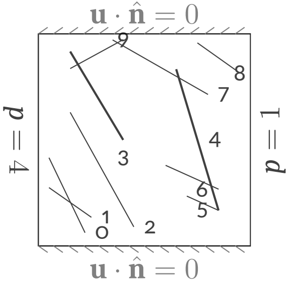
	1030	1043	1050	1128	1758	25697	97987	2341747
1	13	13	12	4	3	36	44	500
2	10	13	12	4	2	36	47	500
4	3	13	12	9	6	31	35	500
8	12	12	10	8	3	21	31	500
16	12	2	29	13	2	36	46	500
32	7	2	3	4	5	55	72	500
48	3	14	13	22	14	19	100	500

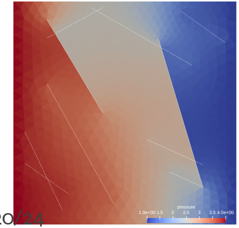
MATCH

	1030	1043	1050	1128	1758	25697	97987	2341747
1	2	8	10	6	4	44	49	500
2	2	2	17	14	10	22	40	356
4	2	7	8	3	10	37	299	187
8	3	4	5	3	6	31	80	196
16	3	11	6	9	3	55	283	113
32	3	5	8	3	2	30	41	178
48	4	10	10	2	6	21	26	287

Algorithmic Scalability (Pure MPI)

4 Few preliminary tests





2D case, several fractures high-difference in permeability.
Operator complexity ($\sum_{l=0}^{N_l-1} \mathrm{nnz}(A_l) / \mathrm{nnz}(A_0) > 1$).

	1030	1043	1050	1128	1758	25697	97987	2341747
1	1.09	1.099	1.094	1.086	1.076	1.064	1.061	1.059
2	1.095	1.103	1.094	1.09	1.082	1.066	1.062	1.059
4	1.094	1.113	1.102	1.101	1.084	1.068	1.062	1.059
8	1.111	1.12	1.116	1.109	1.089	1.071	1.063	1.059
16	1.121	1.134	1.126	1.129	1.099	1.064	1.065	1.06
32	1.158	1.165	1.179	1.14	1.115	1.068	1.06	1.06
48	1.171	1.166	1.178	1.467	1.161	1.071	1.061	1.061

VBM

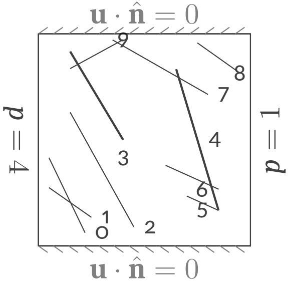
	1030	1043	1050	1128	1758	25697	97987	2341747
1	1.21	1.211	1.209	1.213	1.524	1.303	1.3	1.306
2	1.543	1.534	1.534	1.532	1.219	1.335	1.296	1.305
4	2.408	2.42	2.432	2.431	1.556	1.282	1.309	1.304
8	2.403	2.422	2.424	2.433	2.526	1.36	1.345	1.307
16	2.402	2.421	2.425	2.434	2.526	1.573	1.287	1.306
32	2.404	2.431	2.435	2.429	2.522	1.233	1.365	1.312
48	2.408	2.42	2.411	2.433	2.524	1.622	1.586	1.301

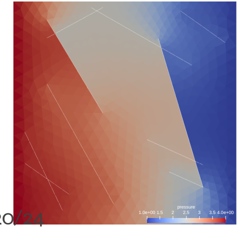
MATCH

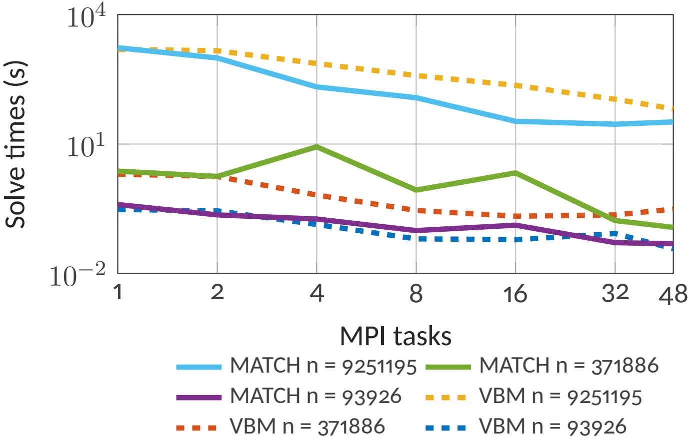
Algorithmic Scalability (Pure MPI)

2D case, several fractures high-difference in permeability.

Solve time.

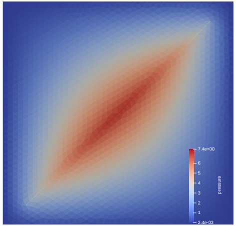


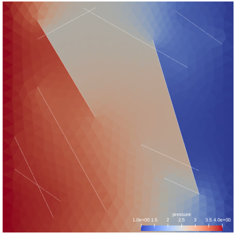




GPU Tests

4 Few preliminary tests





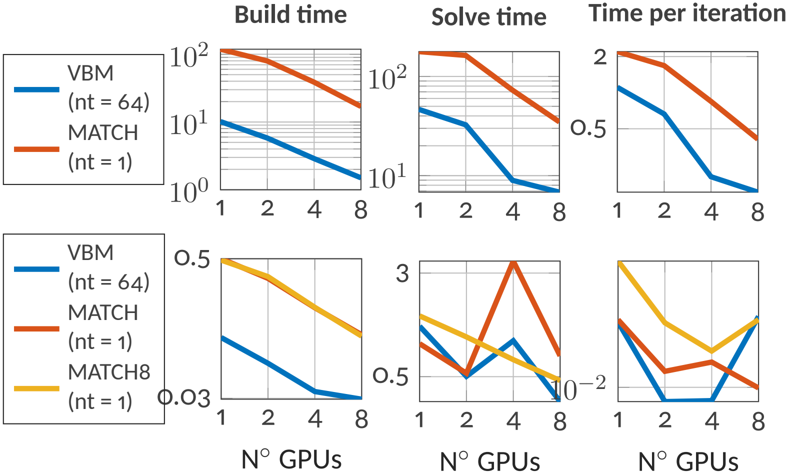


Table of Contents

5 Summary

Conclusions and perspectives

We have shown that. . .

- AMG with ℓ_1-smoothed aggregation is **promising** on the **algorithmic** side,
- **OpenMP acceleration** of **build-phase** routines allows us to better exploit computational resources.

We would like to. . .

- implement an OpenMP version of the *approximate* matching algorithm,
- investigate **specialized smoothers** for the fracture problem:
  - Polynomial acceleration for the non symmetric system?
  - Better weight vector selection for the matching algorithm?

Algebraic Multigrid Methods for Simulation of Single Phase Flow in Fractured Media

Thank you for listening!

Any questions?