Domain decomposition preconditioners for efficient parallel simulations of single-phase flow in three-dimensional fractured porous media with a very large number of fractures

Pierre Jolivet[‡] Michel Kern[*†] Frédéric Nataf[§*] Géraldine Pichot[*†]
Daniel Zegarra Vasquez[*†]

[April 11, 2025 ; first draft, long version]

Abstract

Three-dimensional fractured-porous media are classically described by the Discrete Fracture Matrix (DFM) model, where the rock matrix remains three-dimensional, while the fracture network is of co-dimension one. This study deals with efficient solutions of the linear system arising in single-phase flow in large DFMs using a mixed-hybrid finite element method. It demonstrates the effectiveness of two-level domain decomposition methods based on the GenEO framework for preconditioning iterative solvers applied to such flow problems. Due to the mixed-dimensional nature of the geometry, a specific partitioning strategy is required. The coarse spaces for these preconditioners are built by locally solving spectral problems. This paper presents simulations with very large DFMs, with up to hundreds of thousands of fractures, achieving a low iteration count and short total computation time. Furthermore, this work confirms the scalability of these preconditioners.

Keywords

very large-scale simulations, single-phase flow, heterogeneous fractured porous media, discrete fracture-matrix model, conforming meshes, mixed-hybrid finite element method, gmres, domain decomposition methods, preconditioners.

Introduction

The efficient simulation of single-phase flow in large-scale fractured porous rocks is a challenging computational problem. This paper builds on previous work by some of the authors [49] and aims to enhance computational performance for such simulations. The critical ingredient is the use of a recently developed two-level domain decomposition (DD) preconditioners based on the GenEO framework [43, 45, 44].

In the subsurface, fractures are ubiquitous, appearing as discontinuities in the rock matrix medium in the form of narrow zones. There exists many relevant geometric representations of fractures. In this work, we choose an explicit representation called Discrete Fracture Matrix (DFM) [15, 11, 19], where the fractures are modeled as lower dimensional objects [1, 54, 5]. The rock matrix is a three-dimensional domain and fractures, which, in this work are assumed conductive, are considered as two-dimensional flat surfaces inside a three-dimensional space. The distribution of the fractures follows probability laws derived from field observations [27].

In a DFM representation, Darcy-like averaged equations govern flow in both the rock matrix and the fracture network, with a coupling term that describes exchanges between the two media [54,

[‡]LIP6, CNRS UMR 7606, Sorbonne Université, 4 place Jussieu, 75005 Paris, France.
[*]Inria, 48 rue Barrault, 75013 Paris, France.
[†]CERMICS, École Nationale des Ponts et Chaussées, 6 avenue Blaise Pascal, 77420 Champs-sur-Marne, France.
[§]LJLL, CNRS UMR 7598, Sorbonne Université, 4 place Jussieu, 75005 Paris, France.
E-mails: pierre@joliv.et , michel.kern@inria.fr , frederic.nataf@sorbonne-universite.fr , geraldine.pichot@inria.fr , daniel.zegarra-vasquez@inria.fr .

56]. This results in mixed-dimensional partial differential equations (PDEs), coupling the two-dimensional fracture flow with three-dimensional surrounding flow in the rock matrix.

The mesh of the fractured porous medium is generated in two stages, we first build a surface simplicial mesh of the fracture network, then we generate the volume mesh constrained by the set of fracture faces (*conforming* meshes). Because the fractures often intersect at very small angles, the surface and the volume meshes will usually contain low quality elements [49].

In this work, the set of PDEs is discretized with the mixed-hybrid finite element method (MHFEM) [21, 36, 13, 41] of the lowest order, resulting in a large, sparse, symmetric, and positive definite (SPD) linear system. The matrix of this linear system turns out to be highly ill-conditioned. Two main causes of this ill-conditioning have been identified in [49], namely the presence of low quality elements [25] and strong hydraulic conductivity contrasts. In fact, condition numbers as high as $2 \cdot 10^{16}$ have been observed [49].

The use of black-box solvers has been investigated in [49]. There, it was shown that direct solvers are not viable, as their RAM requirements become prohibitive. Iterative methods such as the conjugate gradient, preconditioned with purely algebraic methods, e.g. Algebraic Multigrid (AMG), were found to display very slow convergence. In the largest cases from [49] with 90k fractures and 39M dofs, more than 200,000 iterations were needed to reach a tolerance of 10^{-12} using CG preconditioned with AMGCL [29, 30, 31]. While there exists other high quality AMG implementations, such as BoomerAMG [42], GAMG [35] or Muelu [55], it has been observed that multigrid methods have difficulties with hybridized discretization [20, 39, 64] and also with meshes containing low quality elements [25]. There exists also ad-hoc methods for the specific case of flow in fractured porous media, such as geometric multigrid (for saddle point problems, which is the case in this work) [7] or approximate block factorization [6].

In this paper, we focus on another family of preconditioners based on DD methods. In the family of DD preconditioners, there exists both single-level preconditioners and two-level (and potentially multi-level) methods [62, 33, 59]. Among single-level preconditioners, with or without overlap, one finds additive Schwarz (ASM), restrictive ASM (RAS) and Neumann-Neumann methods. Two-level methods are needed to obtain scalability when the number of subdomains is increased. Without being exhaustive, we can list MaPHyS [24, 38], BDCC [32, 53], HPDDM based on a splitting method for the coarse space [40], called HPDDM splitting, and HPDDM based on a GenEO coarse space [43, 61, 45, 44], called HPDDM GenEO.

Even if DD methods have been studied for single-phase flow in three-dimensional fractured porous media with a low number of fractures [2, 4, 52, 3], to the best of our knowledge, no study has been carried out with large number of fractures (tens to hundreds of thousands).

In this work, we focus on the additive Schwarz method (ASM) and the restricted additive Schwarz (RAS) as single-level methods and HPDDM (both splitting and GenEO) as two-level methods. The goal of this paper is to assess the performance of these preconditioners by running massively parallel simulations and analyzing the number of iterations, the clock time and the scalability with respect to the number of MPI processes (which is taken equal to the number of subdomains). DD preconditioners require additional information beyond the linear system itself: a partitioning, describing how the unknowns are distributed among the MPI processes. Partitioning can be initiated either from the matrix of the linear system or, more naturally, directly from the geometrical domain mesh. The most widely used partitioning software, such as METIS [26] and SCOTCH [58, 57], converts the input into a connectivity graph, then divide the graph's vertices into subdomains. METIS and SCOTCH both provide high-performance parallel versions, ParMETIS [47] and PT-SCOTCH [26, 57]. In this paper, we utilize METIS and ParMETIS.

The DD methods used in this work are of the overlapping type. On the one hand, the ASM and RAS methods, as well as HPDDM splitting, do not require any geometrical information as the partitioning can be done from the matrix alone. On the other hand, HPDDM GenEO require additional information, in the form of local subdomain Neumann matrices and a global/local

mapping. Providing these data makes it possible to obtain a theoretical guarantee on the number of iterations. Each Neumann matrix must be obtained from a local assembly over its subdomain mesh, with one layer of overlap. In the specific case of DFMs, care must taken to build the overlap both from the porous and the fracture network meshes.

The contributions of this paper are:

- to show the applicability of DD methods to large and very large DFMs;
- to propose an algorithm for partitioning a DFM, based on the mixed-dimensional mesh;
- to generate large (tens of thousands of fractures) and very large (hundreds of thousands of fractures) fractured-porous test cases;
- to show that the two-level HPDDM preconditioner, based on the GenEO framework, outperforms the other tested preconditioners.

The outline of this paper is the following. Section 1 recalls the single-phase flow problem and the corresponding linear system to be solved. Section 2 recalls the theory of DD preconditioners, with a particular focus on the two-level GenEO coarse grid. This section shows that the GenEO framework can be applied to the single-phase flow problem. In particular, it details how to generate fractured-porous subdomains and local Neumann matrices. Section 3 describes the hardware and software context. Section 4 presents the generation of large and very large test cases, featuring up to 697k fractures, 243M dofs and 6825 sub-domains, potentially with strong permeability/transmissivity contrasts. We analyze the performance of DD preconditioners in terms of weak scalability, strong scalability, and the impact of the geometry of the fracture network and extend the test cased presented in [49] to very large-scale parallel simulations. Section 5 contains the results of DD preconditioners, particularly the two-level ones, on the test cases presented in Section 4. We show that simulating single-phase flow in very large three-dimensional fractured porous media can be accomplished with low clock times and small number of iterations with the HPPDM GenEO framework on thousands of cores.

1 Single-phase flow problem mathematical setting

1.1 Discret Fracture Matrix notation

We use the same notation as in [48, 49], which are recalled in this section.

We denote by Ω the domain representing a permeable fractured porous rock. The fracture network $\Omega^{\mathfrak{f}}$ embedded in $\overline{\Omega}$ is modeled as a collection of $N^{\mathfrak{f}}$ fractures represented as polygons: $\Omega^{\mathfrak{f}} = \cup_{\ell=1}^{N^{\mathfrak{f}}} \Omega_\ell^{\mathfrak{f}}$. The rock matrix domain is defined as $\Omega^{\mathfrak{m}} = \Omega \setminus \overline{\Omega^{\mathfrak{f}}}$.

For the purpose of the mathematical description, we divide each polygon $\Omega_\ell^{\mathfrak{f}}, \ell = 1, \ldots, N^{\mathfrak{f}}$, into subpolygons γ_ℓ along each intersection line, such that $\Omega^{\mathfrak{f}} = \cup_{\ell \in \mathcal{I}} \gamma_\ell$, where $\mathcal{I}$ is the index set of the subpolygons. The subpolygons are built so as to satisfy: $\gamma_\ell \cap \gamma_k = \emptyset$ if $\ell \neq k$ and that $\overline{\gamma_\ell} \cap \overline{\gamma_k}$, $\ell \neq k$, is either an edge or a point or an empty set, as described in [63, 48]. The subpolygons γ_ℓ, $\ell \in \mathcal{I}$, further partition the rock matrix domain $\Omega^{\mathfrak{m}}$ into connected components ω_α, such that $\Omega^{\mathfrak{m}} = \cup_{\alpha \in \mathcal{C}} \omega_\alpha$, where $\mathcal{C}$ is the index set of connected components. The subdomains ω_α may also be further subdivided to satisfy: $\omega_\alpha \cap \omega_\beta = \emptyset$ if $\alpha \neq \beta$ and that $\overline{\omega_\alpha} \cap \overline{\omega_\beta}$, $\alpha \neq \beta$, is either a face or an edge or an empty set. Let $\mathcal{I}_\alpha = \{\ell, \gamma_\ell \subset \partial\omega_\alpha\}$ the set of subpolygons on $\partial\omega_\alpha$. We further assume that both the subpolygons and the subdomains are Lipschitz. This is always achievable in practice: for example, after generating a mesh for both $\Omega^{\mathfrak{f}}$ and $\Omega^{\mathfrak{m}}$, the subpolygons and the subdomains may be built by agglomeration of mesh elements.

Dirichlet and Neumann boundary conditions are set on the porous domain on Γ^D and Γ^N respectively, with $\overline{\partial\Omega^{\mathfrak{m}}} = \Gamma_0$ with $\Gamma_0 = \Gamma^D \bigcup \Gamma^N$. Let $\Gamma_\alpha = \partial\omega_\alpha = \Gamma_\alpha^D \cup \Gamma_\alpha^N \cup \Gamma_\alpha^\gamma \cup \Gamma_\alpha^{\text{fict}}$, with $\Gamma_\alpha^D = \partial\omega_\alpha \cap \Gamma^D$, $\Gamma_\alpha^N = \partial\omega_\alpha \cap \Gamma^N$, $\Gamma_\alpha^\gamma = \cup_{\ell \in \mathcal{I}_\alpha} \gamma_\ell$ and $\Gamma_\alpha^{\text{fict}}$ gathers the faces introduced by the

subdivision process (they will only play a role in Section 1.2). We call such a face a"fictitious" face, and note that it belongs to two subdomains.

We assume that Dirichlet and Neumann boundary conditions also hold for the fracture networks on the domain sides: $\overline{\partial\Omega^{\mathfrak{f}}} \cap \overline{\partial\Omega} = \Sigma^D \bigcup \Sigma^N$. For each $\ell \in \mathcal{I}$, $\Sigma_\ell^D = \partial\gamma_\ell \cap \Sigma^D$ and $\Sigma_\ell^N = \partial\gamma_\ell \cap \Sigma^N$. In this work, homogeneous Neumann boundary conditions are applied at the fracture tips (border of the subpolygon γ_ℓ, $\ell \in \mathcal{I}$, that are not intersection and not on the border of the domain). The fracture tips are part of Σ^N.

We denote by $\Sigma_{\ell,0} = \Sigma_\ell^D \cup \Sigma_\ell^N$ the segments shared between the subdomain γ_ℓ and the domain Ω and $\Sigma_0 = \cup_{\ell\in\mathcal{I}}\Sigma_{\ell,0}$. For each $\ell \in \mathcal{I}$, we let $\Sigma_\ell^{\text{inter}}$ be the set of intersections of the sub-polygon γ_ℓ, and we define $\Sigma_\ell = \Sigma_\ell^D \cup \Sigma_\ell^N \cup \Sigma_\ell^{\text{inter}}$. Let Σ be the set of intersections between the sub-polygons. The intersections may be segments shared by two or more sub-polygons. Let $\mathcal{I}_\sigma = \{\ell \in \mathcal{I}, \sigma \subset \Sigma\}$ the index set of polygons that contain σ as intersection.

1.2 Physical governing equations

We consider a single-phase, incompressible, and time-independent flow problem, as detailed for instance in [54, 17, 18, 10, 11, 3, 12]. The unknowns are:

- the fluid flow velocity in the rock matrix domain $u^{\mathfrak{m}} = (u_\alpha^{\mathfrak{m}})_{\alpha\in\mathcal{C}}$, obtained from the fluid velocity $u_\alpha^{\mathfrak{m}}$ computed in each subdomain ω_α, in m/s,
- the fluid flow velocity in the fracture network $u^{\mathfrak{f}} = (u_\ell^{\mathfrak{f}})_{\ell\in\mathcal{I}}$, obtained from the fluid velocity $u_\ell^{\mathfrak{f}}$ computed in each subpolygon γ_ℓ, in m^2/s,
- the hydraulic head in the rock matrix $p^{\mathfrak{m}} = (p_\alpha^{\mathfrak{m}})_{\alpha\in\mathcal{C}}$ obtained from the hydraulic head $p_\alpha^{\mathfrak{m}}$ computed in each subdomain ω_α, in m,
- the hydraulic head in the fracture network $p^{\mathfrak{f}} = (p_\ell^{\mathfrak{f}})_{\ell\in\mathcal{I}}$ obtained from the hydraulic head $p_\ell^{\mathfrak{f}}$ computed in each subdomain γ_ℓ, in m.

In the rock matrix, the hydraulic conductivity is a tensor denoted by $\mathcal{K}^{\mathfrak{m}} = (\mathcal{K}_\alpha^{\mathfrak{m}})_{\alpha\in\mathcal{C}}$. In the fracture network, the tangential hydraulic conductivity is also a tensor denoted $\tilde{\mathcal{K}}^{\mathfrak{f}} = (\tilde{\mathcal{K}}_\ell^{\mathfrak{f}})_{\ell\in\mathcal{I}}$. Their physical unit is m/s. Furthermore, in the fracture network, the aperture is a scalar denoted $d_\ell^{\mathfrak{f}} = (d_\ell^{\mathfrak{f}})_{\ell\in\mathcal{I}}$, whose physical unit is m. Thus, in the fracture network, one can define the transmissivity as a tensor: $\mathcal{K}^{\mathfrak{f}} = (\mathcal{K}_\ell^{\mathfrak{f}})_{\ell\in\mathcal{I}} = (d_\ell^{\mathfrak{f}}\tilde{\mathcal{K}}_\ell^{\mathfrak{f}})_{\ell\in\mathcal{I}}$, whose unit is m^2/s. In the sequel, we only work with $\mathcal{K}^{\mathfrak{f}}$ (no closing of fractures).

The source term in the rock matrix (resp. in the fracture network) is denoted by $f^{\mathfrak{m}} = (f_\alpha^{\mathfrak{m}})_{\alpha\in\mathcal{C}}$ (resp. $f^{\mathfrak{f}} = (f_\ell^{\mathfrak{f}})_{\ell\in\mathcal{I}}$), whose unit is 1/s (resp. (m/s)).

Let $q_\alpha^{\mathfrak{m},N}$ (resp. $q_\ell^{\mathfrak{f},N}$) be the Neumann boundary conditions applying on Γ_α^N (resp. $\Sigma_\ell \cap \Sigma^N$) for all $\alpha \in \mathcal{C}$ (resp. for all $\ell \in \mathcal{I}$). The Neumann condition in the rock matrix (resp. in the fracture network) is denoted $q^{\mathfrak{m},N} = (q_\alpha^{\mathfrak{m},N})_{\alpha\in\mathcal{C}}$ (resp. $q^{\mathfrak{f},N} = (q_\ell^{\mathfrak{f},N})_{\ell\in\mathcal{I}}$), whose unit is m/s (resp. (m^2/s)).

Let $g_\alpha^{\mathfrak{m},D}$ (resp. $g_\ell^{\mathfrak{f},D}$) be the Dirichlet boundary conditions applying on Γ_α^D (resp. $\Sigma_\ell \cap \Sigma^D$), for all $\alpha \in \mathcal{C}$ (resp. for all $\ell \in \mathcal{I}$). The Dirichlet condition in the rock matrix (resp. in the fracture network) is denoted by $g^{\mathfrak{m},D} = (g_\alpha^{\mathfrak{m},D})_{\alpha\in\mathcal{C}}$ (resp. $g^{\mathfrak{f},D} = (g_\ell^{\mathfrak{f},D})_{\ell\in\mathcal{I}}$), whose unit is m (resp. (m)).

For a smooth function $p^{\mathfrak{m}}$ (resp. $p_\alpha^{\mathfrak{m}}$) defined in $\Omega^{\mathfrak{m}}$ (resp. a subdomain ω_α), we let $\gamma^{\mathfrak{m}}p^{\mathfrak{m}}$ (resp. $\gamma^{\mathfrak{m}}p_\alpha^{\mathfrak{m}}$) denote the restriction of $p^{\mathfrak{m}}$ to the boundary of $\Omega^{\mathfrak{m}}$ (resp. ω_α). For a smooth vector field $u^{\mathfrak{m}}$ defined in $\Omega^{\mathfrak{m}}$, we let $\gamma_n^{\mathfrak{m}}u^{\mathfrak{m}}$ denote the normal trace of $u^{\mathfrak{m}}$ along (a part of) the boundary of $\Omega^{\mathfrak{m}}$. Informally, $\gamma_n^{\mathfrak{m}}u^{\mathfrak{m}} = u^{\mathfrak{m}} \cdot \nu_{|\partial\Omega^{\mathfrak{m}}}$, where ν is the unit outward normal to $\partial\Omega^{\mathfrak{m}}$. If we want to specify the subdomain ω_α over which $u_\alpha^{\mathfrak{m}}$ is defined, we write $\gamma_n^{\mathfrak{m}}u_\alpha^{\mathfrak{m}}$. If the part of the boundary is an immersed polygon $\gamma_\ell, \ell \in \mathcal{I}$, we write $\gamma_n^{\mathfrak{m},\pm}u_{\alpha_{\ell\pm}}^{\mathfrak{m}}$ depending on which side of the polygon $\omega_{\alpha\ell^\pm}$ we are looking at.

Similarly for a smooth scalar function $p^{\mathfrak{f}}$ or a smooth (tangential) vector field $u^{\mathfrak{f}}$ defined on $\Omega^{\mathfrak{f}}$ (resp. $p_\ell^{\mathfrak{f}}$ or $u_\ell^{\mathfrak{f}}$ defined over a polygon γ_ℓ), we denote by $\gamma^{\mathfrak{f}}p$ and $\gamma_n^{\mathfrak{f}}u^{\mathfrak{f}}$ (resp. $\gamma^{\mathfrak{f}}p_\ell$ and $\gamma_n^{\mathfrak{f}}u_\ell^{\mathfrak{f}}$)

its (normal) trace along the boundary of $\Omega^{\mathfrak{f}}$ (resp. of γ_ℓ). If we are looking at the trace along a particular intersection segment σ, we denote the (normal) trace by $\gamma^{\mathfrak{f},\sigma} p_\ell$ and $\gamma_n^{\mathfrak{f},\sigma} u_\ell^{\mathfrak{f}}$.

For a face γ shared by two subdomains α^+ and α^-, we define the jump across γ as

$$[[u^{\mathfrak{m}}]]_{|\gamma} = \gamma_n^{m,+} u_{\alpha^+}^{\mathfrak{m}} + \gamma^{m,-} u_{\alpha^-}^{\mathfrak{m}}. \tag{1.1}$$

In each porous subdomain ω_α, $\alpha \in \mathcal{C}$, the flow problem is written as follows:

$$\nabla \cdot u_\alpha^{\mathfrak{m}} = f_\alpha^{\mathfrak{m}} \text{ in } \omega_\alpha, \tag{1.2a}$$
$$u_\alpha^{\mathfrak{m}} = -\mathcal{K}_\alpha^{\mathfrak{m}} \nabla p_\alpha^{\mathfrak{m}} \text{ in } \omega_\alpha, \tag{1.2b}$$
$$\gamma^{\mathfrak{m}} p_\alpha^{\mathfrak{m}} = g_\alpha^{\mathfrak{m},D} \text{ on } \Gamma_\alpha^D, \tag{1.2c}$$
$$\gamma_n^{\mathfrak{m}} u^{\mathfrak{m}} = q_\alpha^{\mathfrak{m},N} \text{ on } \Gamma_\alpha^N. \tag{1.2d}$$

Additionally, across a fictitious face shared by two subdomains both the hydraulic head and the jump of the flux is zero.

In each polygon γ_ℓ, $\ell \in \mathcal{I}$, the flow problem is written as follows:

$$\nabla_\tau \cdot u_\ell^{\mathfrak{f}} = [[u^{\mathfrak{m}}]]_{|\gamma_\ell} + f_\ell^{\mathfrak{f}} \text{ in } \gamma_\ell, \tag{1.3a}$$
$$u_\ell^{\mathfrak{f}} = -\mathcal{K}_\ell^{\mathfrak{f}} \nabla_\tau p_\ell^{\mathfrak{f}} \text{ in } \gamma_\ell, \tag{1.3b}$$
$$\gamma^{\mathfrak{f}} p_\ell^{\mathfrak{f}} = g_\ell^{\mathfrak{f},D} \text{ on } \Sigma_\ell^D, \tag{1.3c}$$
$$\gamma_n^{\mathfrak{f}} u_\ell^{\mathfrak{f}} = q_\ell^{\mathfrak{f},N} \text{ on } \Sigma_\ell^N, \tag{1.3d}$$

where $[[u^{\mathfrak{m}}]]_{|\gamma_\ell}$ is an exchange term that couples the flow between the porous subdomains $\omega_{\alpha_\ell^+}$ and $\omega_{\alpha_\ell^-}$ and the fracture $\gamma_\ell = \overline{\omega_{\alpha_\ell^+}} \cap \overline{\omega_{\alpha_\ell^-}}$. This term acts as a source term for the fracture flow problem and is defined as follows:

$$[[u^{\mathfrak{m}}]]_{|\gamma_\ell} = \gamma_n^{\mathfrak{m},+} u_{\alpha_\ell^+}^{\mathfrak{m}} + \gamma^{\mathfrak{m},-} u_{\alpha_\ell^-}^{\mathfrak{m}}, \forall \ell \in \mathcal{I}. \tag{1.4}$$

We assume the continuity of the hydraulic heads between the subdomains and the polygons [5]:

$$\gamma^{\mathfrak{m}} p_{\alpha_\ell^+}^{\mathfrak{m}} = \gamma^{\mathfrak{m}} p_{\alpha_\ell^-}^{\mathfrak{m}} = p_\ell^{\mathfrak{f}}, \forall \ell \in \mathcal{I}. \tag{1.5}$$

We also add conditions at each fracture-fracture intersection $\sigma \in \Sigma$ [63]:

$$\gamma^{\mathfrak{f},\sigma} p_\ell^{\mathfrak{f}}|_\sigma = \gamma^{\mathfrak{f},\sigma} p_k^{\mathfrak{f}}|_\sigma \text{ for } (\ell, k) \in \mathcal{I}_\sigma^2, \forall \sigma \subset \Sigma, \tag{1.6a}$$
$$\sum_{\ell \in \mathcal{I}_\sigma} \gamma_n^{\mathfrak{f},\sigma} u_\ell^{\mathfrak{f}} = 0, \text{ for each } \sigma \subset \Sigma. \tag{1.6b}$$

Condition (1.6a) represents the continuity of the hydraulic head $p^{\mathfrak{f}}$ for each intersection $\sigma \subset \Sigma$, while (1.6b) represents the mass balance of the fracture fluid's flux per unit length across each intersection $\sigma \subset \Sigma$ between fractures.

We assume that $\mathcal{K}^{\mathfrak{m}}(x)$ (resp. $\mathcal{K}^{\mathfrak{f}}(x)$) is a uniformly symmetric and positive definite tensor so that there exists constants $0 < K_{\min}^{\mathfrak{m}}, K_{\max}^{\mathfrak{m}} < +\infty$ (resp. $0 < K_{\min}^{\mathfrak{f}}, K_{\max}^{\mathfrak{f}} < +\infty$) such that:

$$K_{\min}^{\mathfrak{m}} \|\xi\|_2^2 \leq \xi^T \mathcal{K}^{\mathfrak{m}}(x) \xi \leq K_{\max}^{\mathfrak{m}} \|\xi\|_2^2, \qquad \forall \xi \in \mathbb{R}^3, \ \forall x \in \Omega^{\mathfrak{m}}, \tag{1.7a}$$
$$K_{\min}^{\mathfrak{f}} \|\xi\|_2^2 \leq \xi^T \mathcal{K}^{\mathfrak{f}}(x) \xi \leq K_{\max}^{\mathfrak{f}} \|\xi\|_2^2, \qquad \forall \xi \in \mathbb{R}^2, \ \forall x \in \Omega^{\mathfrak{f}}. \tag{1.7b}$$

We further assume that the source terms $f_\alpha^{\mathfrak{m}}$ (resp. $f_\ell^{\mathfrak{f}}$) live in $L^2(\Omega_\alpha^{\mathfrak{m}})$ (resp. $L^2(\gamma_\ell^{\mathfrak{f}})$), that the Dirichlet boundary conditions functions $g_\alpha^{\mathfrak{m},D}$ (resp. $g_\ell^{\mathfrak{f},D}$) are in $H^{\frac{1}{2}}(\Gamma_\alpha^D)$ (resp. $H^{\frac{1}{2}}(\Sigma_\ell^D)$) and that the Neumann boundary conditions functions $q_\alpha^{\mathfrak{m},N}$ (resp. $q_\alpha^{\mathfrak{f},N}$) are in $H^{-\frac{1}{2}}(\Gamma_\alpha^N)$ (resp. $H^{-\frac{1}{2}}(\Sigma_\ell^N)$).

1.3 Linear system and algebraic bilinear form

According to [48], we obtain both a mesh-dependant algebraic bilinear form and a matrix formulation of the mixed-hybrid weak formulation after choosing the bases and the degrees of freedom for the involved spaces.

- The spaces for the velocities $V_h^{\mathrm{hyb},\mathfrak{m}}$ and $V_h^{\mathrm{hyb},\mathfrak{f}}$, and thus V_h^{hyb}, is provided with the natural bases obtained from the product of the local Raviart-Thomas(-Nédélec) spaces. Their degrees of freedom are the fluxes of each face of each tetrahedron (resp. each edge of each triangle) for the matrix and the fractures respectively:

$$\boldsymbol{Q}^{\mathfrak{m}} = (q_{T,f}^{\mathfrak{m}})_{(T\in\mathcal{T}_h, f\in\{1,2,3,4\})} \in \mathbb{R}^{4\mathcal{N}(\mathcal{T}_h)}, \quad \boldsymbol{Q}^{\mathfrak{f}} = (q_{K,e}^{\mathfrak{f}})_{(K\in\mathcal{K}_h, e\in\{1,2,3\})} \in \mathbb{R}^{3\mathcal{N}(\mathcal{K}_h)};$$

- The spaces for the hydraulic head are piecewise constant functions over each tetrahedron or triangles respectively:

$$\boldsymbol{P}^{\mathfrak{m}} = (p_T^{\mathfrak{m}})_{T\in\mathcal{T}_h} \in \mathbb{R}^{\mathcal{N}(\mathcal{T}_h)}, \quad \boldsymbol{P}^{\mathfrak{f}} = (p_K^{\mathfrak{f}})_{K\in\mathcal{K}_h} \in \mathbb{R}^{\mathcal{N}(\mathcal{K}_h)};$$

- The spaces for the Lagrange multipliers, also known as traces of hydraulic head, are piecewise constant on the inner faces of the tetrahedral mesh, and on the edges of the triangular mesh respectively:

$$\boldsymbol{\Lambda}^{\mathfrak{m}} = (\lambda_F^{\mathfrak{m}})_{F\in\mathcal{E}_h^{\mathrm{in}}} \in \mathbb{R}^{\mathcal{N}(\mathcal{F}_h^{\mathrm{in},N})}, \quad \boldsymbol{\Lambda}^{\mathfrak{f}} = (\lambda_E^{\mathfrak{f}})_{E\in\mathcal{N}(\mathcal{E}_h\setminus\mathcal{E}_h^D)} \in \mathbb{R}^{\mathcal{N}(\mathcal{E}_h\setminus\mathcal{E}_h^D)}.$$

Thus, the flux unknown is denoted $\boldsymbol{Q} = (\boldsymbol{Q}^{\mathfrak{m}}, \boldsymbol{Q}^{\mathfrak{f}})$, the hydraulic head unknown is denoted $\boldsymbol{P} = (\boldsymbol{P}^{\mathfrak{m}}, \boldsymbol{P}^{\mathfrak{f}})$ and the trace of hydraulic head unknown is denoted $\boldsymbol{\Lambda} = (\boldsymbol{\Lambda}^{\mathfrak{m}}, \boldsymbol{\Lambda}^{\mathfrak{f}})$. The mixed-hybrid variational discrete formulation can be written in its algebraic form:

$$\begin{pmatrix} \mathcal{A} & -\mathcal{B} & \mathcal{C} \\ \mathcal{B}^T & 0 & 0 \\ \mathcal{C}^T & 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{Q} \\ \boldsymbol{P} \\ \boldsymbol{\Lambda} \end{pmatrix} = \begin{pmatrix} L_Q \\ L_P \\ 0 \end{pmatrix}, \tag{1.8}$$

where L_Q (resp. L_P) is the algebraic version of the linear operator L_a (resp. L_b) presented in [48], and where the three matrices $\mathcal{A}, \mathcal{B}$ and $\mathcal{C}$ are block 2×2 matrices, with the following structure:

$$\mathcal{A} = \begin{pmatrix} A^{\mathfrak{m}} & 0 \\ 0 & A^{\mathfrak{f}} \end{pmatrix}, \quad \mathcal{B} = \begin{pmatrix} B^{\mathfrak{m}} & B^{\mathfrak{m},\mathfrak{f}} \\ 0 & B^{\mathfrak{f}} \end{pmatrix}, \quad \mathcal{C} = \begin{pmatrix} C^{\mathfrak{m}} & 0 \\ 0 & C^{\mathfrak{f}} \end{pmatrix}. \tag{1.9}$$

The definition of these blocks is detailed in [48]. As usual for hybridized formulations, it is possible to eliminate the flux unknown $\boldsymbol{Q}$ at the element level. After the global linear system (1.11) is solved, we can compute these fluxes. An illustration of the degrees of freedom $\boldsymbol{P}$ and $\boldsymbol{\Lambda}$ with two tetrahedral elements T_1 and T_2 (rock) and one triangle K (fracture) is displayed on Figure 1).

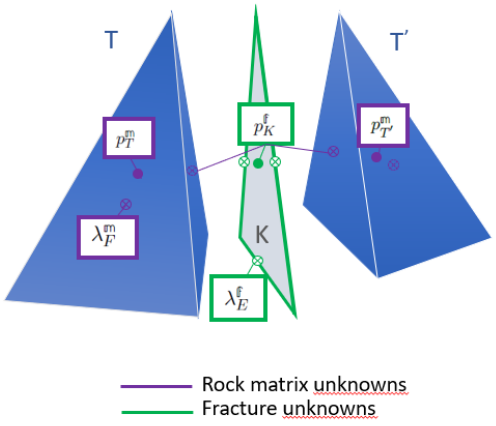


Figure 1: Mixed-hybrid finite element unknowns

On the one hand, for the matrix formulation, after reordering the unknowns to group matrix and fracture unknowns together, one obtains a reduced linear system for the pressure and multiplier unknowns:

$$\begin{pmatrix} D^{\mathfrak{m}} & -R^{\mathfrak{m},\mathfrak{m}} & -R^{\mathfrak{m},\mathfrak{f}} & 0 \\ -(R^{\mathfrak{m},\mathfrak{m}})^T & M^{\mathfrak{m},\mathfrak{m}} & M^{\mathfrak{m},\mathfrak{f}} & 0 \\ -(R^{\mathfrak{m},\mathfrak{f}})^T & (M^{\mathfrak{m},\mathfrak{f}})^T & M^{\mathfrak{f},\mathfrak{f}} + D^{\mathfrak{f}} & -R^{\mathfrak{f}} \\ 0 & 0 & -(R^{\mathfrak{f}})^T & M^{\mathfrak{f}} \end{pmatrix} \begin{pmatrix} \boldsymbol{P}^{\mathfrak{m}} \\ \boldsymbol{\Lambda}^{\mathfrak{m}} \\ \boldsymbol{P}^{\mathfrak{f}} \\ \boldsymbol{\Lambda}^{\mathfrak{f}} \end{pmatrix} = \begin{pmatrix} \mathscr{F}^{\mathfrak{m}} \\ \mathscr{V}^{\mathfrak{m}} \\ \mathscr{V}^{\mathfrak{m},\mathfrak{f}} + \mathscr{F}^{\mathfrak{f}} \\ \mathscr{V}^{\mathfrak{f}} \end{pmatrix}. \tag{1.10}$$

The definition of each block and its respective size is given in appendix A.1.1. In [48] it is shown that the left hand side of (1.10) is symmetric, positive and definite, thus, this algebraic problem has a unique solution.

Furthermore, the matrix $D^{\mathfrak{m}}$ is again diagonal, so that the unknowns $\boldsymbol{P}^{\mathfrak{m}}$ can be eliminated at the element level thanks to a Schur complement, which preserves the well-posedness of the algebraic problem (1.10):

$$\begin{pmatrix} A^{\mathfrak{m},\mathfrak{m}} & A^{\mathfrak{m},\mathfrak{f}} & 0 \\ (A^{\mathfrak{m},\mathfrak{f}})^T & A^{\mathfrak{f},\mathfrak{f}} + D^{\mathfrak{f}} & -R^{\mathfrak{f}} \\ 0 & -(R^{\mathfrak{f}})^T & M^{\mathfrak{f}} \end{pmatrix} \begin{pmatrix} \boldsymbol{\Lambda}^{\mathfrak{m}} \\ \boldsymbol{P}^{\mathfrak{f}} \\ \boldsymbol{\Lambda}^{\mathfrak{f}} \end{pmatrix} = \begin{pmatrix} \tilde{\mathscr{F}}^{\mathfrak{m}} + \mathscr{V}^{\mathfrak{m}} \\ \tilde{\mathscr{F}}^{\mathfrak{m},\mathfrak{f}} + V^{\mathfrak{m},\mathfrak{f}} + \mathscr{F}^{\mathfrak{f}} \\ \mathscr{V}^{\mathfrak{f}} \end{pmatrix} \tag{1.11}$$

The definition of each block and its respective size is given in appendix A.1.2. Hereafter, when we refer to "the linear system", we refer to (1.11).

2 Domain decomposition preconditioners

As seen in [49], when solving the linear system (1.11) with a large number of fractures $N^{\mathfrak{f}}$, iterative linear solvers, such as the conjugate gradient (CG) preconditioned with algebraic methods, struggle in terms of clock time and number of iterations. This is why we want to build a preconditioner that is efficient with respect to these two quantities of interest. Thus, we explore the applicability of domain decomposition preconditioners for the single-phase flow in three-dimensional fractured porous media. In particular, we explore the high-performance unified framework for domain decomposition methods, also known as "HPDDM" [43, 33, 45, 44], a two-level domain decomposition preconditioner based on GenEO (and alternatively splitting-based [40]) coarse spaces.

2.1 HPDDM theoretical framework

The HPDDM theoretical framework comes from [43, 45, 44], and a more generic presentation of domain decomposition preconditioners can be found in [33].

2.1.1 Partition of unity

We consider a symmetric, definite and positive linear system $Ax = b$. The first needed ingredient is the partition of unity. Let us recall Definition 1.11 in [33].

Definition 1 (Algebraic partition of unity)**.** *At the discrete level, the main ingredients of the partition of unity are:*

- *a set of indices of degrees of freedom $\mathcal{I}^{dofs}$ and a decomposition into N subsets $\mathcal{I}^{dofs} = \cup_{i=1}^{N} \mathcal{I}_i^{dofs}$;*
- *a vector $U \in \mathbb{R}^{\mathcal{N}(\mathcal{I}^{dofs})}$;*
- *the restriction of a vector $U \in \mathbb{R}^{\mathcal{N}(\mathcal{I}^{dofs})}$ to a subdomain Ω_i, $1 \leq i \leq N$, which can be expressed as $R_i U$, where R_i is a rectangular $\mathcal{N}(\mathcal{I}_i^{dofs}) \times \mathcal{N}(\mathcal{I}^{dofs})$ Boolean matrix; the extension operator is its transpose R_i^T;*

- *the partition of unity at the discrete level which correspond to diagonal matrices D_i of size $\mathcal{N}(\mathcal{I}_i^{dofs}) \times \mathcal{N}(\mathcal{I}_i^{dofs})$ with non-negative entries such that for all vectors $U \in \mathbb{R}^{\mathcal{N}(\mathcal{I}^{dofs})}$*

$$U = \sum_{i=1}^{N} R_i^T D_i R_i U,$$

or, in other words,

$$Id_{\mathcal{N}(\mathcal{I}^{dofs})} = \sum_{i=1}^{N} R_i^T D_i R_i, \tag{2.1}$$

where $Id_{\mathcal{N}(\mathcal{I}^{dofs})} \in \mathbb{R}^{\mathcal{N}(\mathcal{I}^{dofs}) \times \mathcal{N}(\mathcal{I}^{dofs})}$ is the identity matrix of size $\mathcal{N}(\mathcal{I}^{dofs})$.

The upper-script $\delta = 1$ indicates one layer of overlap. For the sake of simplicity (and by convenience for GenEO theory, explained later in Section 2.1.3), we stick to one-layer overlapping subdomains. Each subdomain, including the overlap, is denoted Ω_i^δ, and its interior without the overlap is denoted Ω_i.

In practice, such subdomains can be created thanks to specialized partitioning software (see section 2.2.2 for details). There are two ways to create a partitioning of Ω:

- geometrically, the most natural, thanks to $\mathcal{T}_h$;
- algebraically, directly thanks to A.

In both cases, we consider the set of indices $\mathcal{I}_i^{\text{dofs},\delta=1}$, $i = \{1, \ldots, N\}$ to define the partition of unity. Therefore the matrix R_i is of size $\mathcal{N}(\mathcal{I}_i^{\text{dofs},\delta=1}) \times \mathcal{N}(\mathcal{I}^{\text{dofs}})$. Note that in the PETSc implementation of HPDDM [44], the definition of the partition of unity is a bit different, so called a *Boolean* partition of unity [23]. It consists in defining matrices $\widetilde{R}_i$ such that:

$$\sum_{i=1}^{N} R_i^T \widetilde{R}_i R_i^T R_i = Id_{\mathcal{I}^{\text{dofs}}}.$$

To do so, we need to define the diagonal matrices $\widetilde{D}_i$, $i = 1, \ldots, N$ whose elements on the diagonal element is set to 1 for all inner dofs that are in $\mathcal{I}_i^{\text{dofs}}$ and to 0 to the dofs on the overlap. Then we define the matrices $\widetilde{R}_i = \widetilde{D}_i R_i$, $i = \{1, \ldots, N\}$. Note that the matrices $\{\widetilde{R}_i\}_{i=1}^N$ are the same operators as $\{R_i\}_{i=1}^N$ except that coefficients on the overlap are set to 0 and $\widetilde{R}_i R_i^T = \widetilde{D}_i$.

2.1.2 One-level domain decomposition methods

We start with one-level domain decomposition methods [33].

In the Additive Schwarz Method (ASM) each subdomain has its own local problem. These local problems are solved independently, and the solutions are combined to form the preconditioner M_{ASM}^{-1}:

$$M_{\text{ASM}}^{-1} = \sum_{i=1}^{N} R_i^T (A_i)^{-1} R_i, \tag{2.2}$$

where A_i is defined by $A_i = R_i A R_i^T$.

The Restricted Additive Schwarz (RAS) preconditioner [23] is a non-symmetric version of ASM preconditioner where local subdomain solutions are restricted to the non-overlapping parts of the domain when forming the global solution:

$$M_{\text{RAS}}^{-1} = \sum_{i=1}^{N} \widetilde{R}_i^T (A_i)^{-1} R_i. \tag{2.3}$$

It is known that RAS improves performance with respect to the number of iterations if we compare it to ASM [34], but at the cost of losing symmetry. Thus, such a preconditioner cannot be applied to symmetric iterative solvers such as CG, but can be used with non-symmetric solvers such as GMRES or f-GMRES.

2.1.3 Two-level domain decomposition methods based on spectral coarse grids

The efficiency of one-level domain decomposition preconditioners is known to deteriorate as the number of subdomains increases [33]. In order to make the domain decomposition more scalable, a key approach is to add a coarse grid correction to ASM or RAS. In a two-level method, the solution process is split into two components:

- a fine grid level, where local solutions are computed on overlapping subdomains thanks to ASM or RAS (see above), leading to a preconditioner component denoted M_{fine}^{-1} ;
- a coarse grid correction, where a global correction ensures faster convergence, especially for low-frequency error components that cannot be easily resolved by local subdomain solvers, leading to a preconditioner component denoted M_{coarse}^{-1}.

In this section, we choose ASM in the fine level, since the theoretical results presented hereafter have been proven only with this method. In practice, RAS leads to smaller iteration counts, but then without theoretical control. Thus, in this section, $M_{\text{fine}}^{-1} = M_{\text{ASM}}^{-1}$, please refer to equation (2.2).

One of the most efficient way to build the coarse grid is by using spectral information, as demonstrated in the GenEO framework [43, 61, 45, 44], the Generalized Eigenvalue problems on the Overlap. The GEVP solved by GenEO requires going back to the geometric domain Ω. The GEVP needs a partition of Ω in subdomains Ω_i, $i = \{1, \ldots, N\}$, such that $\Omega = \cup_{i=1,\ldots,N}\Omega_i$ and for each Ω_i, an extension to a domain Ω_i^δ by adding one layer of overlap. The matrix R_i, $i = \{1, \ldots, N\}$, in Equation (2.1), corresponds to the restriction to subdomain Ω_i^δ. These subdomains Ω_i^δ, $i = \{1, \ldots, N\}$, are used to build local Neumann matrices A_i^δ, by assembling the flow problem on Ω_i^δ, which corresponds to a problem with Neumann boundary conditions.

The eigenvectors of the GEVP are then gathered to form a coarse basis that captures the difficult-to-resolve components of the global problem by selecting the eigenfunctions corresponding to the smallest eigenvalues to form the coarse space. By addressing these components with such a coarse space, the fine-level domain decomposition preconditioner performs more efficiently.

Since GenEO requires the Neumann matrices, some geometric information is typically needed, and it cannot be implemented solely with access to the global matrix A in the standard approach. While alternative algebraic strategies might exist to extend GenEO without explicit geometric partitioning, they are not considered in this work.

We describe the complete workflow in order to build the GenEO coarse space (details can be found in [44]):

Algorithm 1 Construction of the spectral coarse grid (from [44]).

1: If we consider an elliptic PDE on Ω, obtain the corresponding mesh $\mathcal{T}_h$ then partition Ω geometrically into N one-layer overlapping subdomains Ω_i, thus obtaining R_i and $\widetilde{R}_i$ restriction operators, where $i \in [\![1, N]\!]$.
2: Assemble the global linear system $Ax = b$.
3: Build the Neumann matrices A_i^δ on Ω_i. The local matrices A_i can be easily obtained thanks to A and R_i.
4: Choose a threshold parameter $\nu \in]0, 1[$ useful for generalized eigenvalue problems (GEVP).
5: **for** $i \leftarrow 1$ to N **do**:
6: Solve a local GEVP on subdomain Ω_i. In other words, find the $\mu_i \in \mathbb{N}^*$ smallest eigenpairs $\{y_{i,j}, \rho_{i,j}\}_{j=1}^{\mu_i}$ up to the desired tolerance ν (i.e., $\rho_{i,j} \leq \nu$) such that:

$$A_i^\delta y_{i,j} = \rho_{i,j} \widetilde{R}_i (R_i)^T A_i R_i (\widetilde{R}_i)^T y_{i,j}. \tag{2.4}$$

7: Assemble a local deflation dense matrix W_i on subdomain Ω_i:

$$W_i = \left[\widetilde{R}_i (R_i)^T y_{i,1} \quad \dots \quad \widetilde{R}_i (R_i)^T y_{i,\mu_i} \right].$$

8: **end for**
9: Compute the global deflation matrix P:

$$P = \left[(R_1)^T W_1 \quad \dots \quad (R_N)^T W_N \right].$$

10: Compute the GenEO coarse operator M_{coarse}^{-1} thanks to a Galerkin product between A and P:

$$M_{\text{coarse}}^{-1} = P (P^T A P)^{-1} P^T.$$

The final two-level domain decomposition preconditioner called "HPDDM GenEO", is obtained by combining the fine and coarse operators. There are three possibilities: additive, deflated or balanced.

$$M_{\text{additive}}^{-1} = M_{\text{coarse}}^{-1} + M_{\text{fine}}^{-1}, \tag{2.5a}$$
$$M_{\text{deflated}}^{-1} = M_{\text{coarse}}^{-1} + M_{\text{fine}}^{-1} (I - A M_{\text{coarse}}^{-1}), \tag{2.5b}$$
$$M_{\text{balanced}}^{-1} = M_{\text{coarse}}^{-1} + (I - M_{\text{coarse}}^{-1} A) M_{\text{fine}}^{-1} (I - A M_{\text{coarse}}^{-1}). \tag{2.5c}$$

We have the following result on the condition number of the preconditioned version of the linear system $Ax = b$.

Theorem 2.1 ([33, Theorem 7.4.2])**.** *The coarse space GenEO guarantees that the condition number κ of the two-level preconditioned operator $M_{additive}^{-1} A$ (see equation* (2.5a)*) is bounded from above:*

$$\kappa(M_{additive}^{-1} A) \leq 2k_0 (2 + (2k_0 + 1) k_1 \nu^{-1}), \tag{2.6}$$

where $k_0 \in \mathbb{N}^*$ is the maximum number of neighbours of a subdomain including itself, $k_1 \in \mathbb{N}^*$ is the maximum multiplicity of a degree of freedom and $\nu \in]0, 1[$ is the user-chosen parameter for the GEVP (2.4)*.*

Remark 2.2. *The parameter ν lets the user control the trade-off between the cost and quality of the preconditioner. The inequality 2.6 shows that a small value of ν means that few eigenvalues will be computed and the condition number may increase, while a larger value of ν means that more eigenvalues will be computed, likely increasing the quality of the preconditioner.*

Remark 2.3. *The proof of this theorem in [33] is fully algebraic except for Lemma 7.4.4, which depends on the existence of a variational formulation of the discrete problem with a bilinear form that is additive and local. The is the only condition that needs to be checked before applying GenEO in a specific case. See Section 2.2.1 for our application to fractured porous flow.*

There exist another version of this two-level domain decomposition preconditioner which does not require the Neumann matrices A_i^δ. It is a local block splitting method using lumping in the overlap, as described in detail in [40], hereafter called HPDDM splitting. The advantage of this method is that it creates a coarse space, thus helping the fine-level part of the preconditioner, as GenEO would. The main difference is that the left-hand side operator replacing A_i^δ in equation (2.4) does not necessarily require any geometrical information to build it. Thus, an algebraic partitioning of A is enough to obtain the two-level preconditioner, following similar steps from GenEO.

Nonetheless, the construction of such an algebraic operator does not guarantee a sharp bound on the condition number of the preconditioned linear system, unlike GenEO with Neumann matrices. A theorem similar to Theorem 2.1 is proved in [40] under the hypothesis that A is diagonally dominant.

2.2 Application to single-phase flow in fractured porous media

In this section, we go back to the linear system (1.11) arising in our application, and we denote this linear system by $Ax = b$. The matrix A is very large, sparse, symmetric and positive definite. We aim at solving the system with Krylov method preconditioned thanks to DD methods. When we use algebraic methods such as single-level ASM and RAS, and also HPDDM splitting, one could choose between partitioning Ω and assembling the local matrices or partitioning the full matrix A. In this work, because it leads to a simpler implementation, we only partition A. Nonetheless, such preconditioners are not scalable with respect to the number of subdomains. We recall that in [49] it has been shown that the linear system (1.11) is very large and very ill-conditioned.

In order to use HPDDM GenEO in the context of fractured porous flow, there are specific steps to be performed. First, it is required to check that the condition explained in Remark 2.3 is fulfilled. This is done in Section 2.2.1. Second, HPDDM GenEO needs Neumann matrices for the GEVP (2.4). The main difficulty is not the assembly phase itself, since it is very close to what is done for the global problem. The key difficulty is to create a coherent partitioning with overlap. Indeed, in our case, Ω is made of two different components: $\Omega^{\mathbb{f}}$, meshed conformingly with triangles $\mathcal{K}_h$, and $\Omega^{\mathbb{m}}$, meshed conformingly with tetrahedra $\mathcal{T}_h$ taking $\mathcal{K}_h$ as constraints. Even if the triangles in $\mathcal{K}_h$ are essentially "special" faces from the tetrahedra in $\mathcal{T}_h$, there are many particular configurations we have to consider in order to successfully obtain subdomains with a one-layer overlap. One of the main contributions of this work is to give an algorithm that describes subdomain generation with a one-layer overlap for three-dimensional Discrete Fracture Matrix (DFM).

2.2.1 Compatibility of the discrete bilinear form

The discrete bilinear form of the fractured porous single-phase flow problem is mesh-dependent. Its construction has been explained thouroughly in [48]. In this section, we recall the main results and we show the compatibility with "HPDDM Neumann" theoretical framework from section 2.1.

If we want to benefit from bounds on the conditioning result from Theorem 2.1, the mathematical objects we use in this work should satisfy the conditions from the lemma mentioned in remark 2.3. In our case, the discrete bilinear form $a_\mathcal{T}$ and the discrete linear form $L_\mathcal{T}$ are the following:

$$a_\mathcal{T}(\tilde{\Lambda}, \hat{\Lambda}) = L_\mathcal{T}(\hat{\Lambda}), \quad \forall \hat{\Lambda} = (\hat{\lambda}^{\mathbb{m}}, \hat{p}^{\mathbb{f}}, \hat{\lambda}^{\mathbb{f}}), \tag{2.7}$$

where:

$$a_{\mathcal{T}}(\tilde{\Lambda},\hat{\Lambda}) = (\hat{q}_\lambda^{\mathfrak{m}})^T A^{\mathfrak{m}}(q_\lambda^{\mathfrak{m}} + q_p^{\mathfrak{m}}) + (\hat{q}_p^{\mathfrak{m}})^T A^{m}(q_\lambda^{\mathfrak{m}} + q_p^{\mathfrak{m}}) + (\hat{q}_p^{\mathfrak{f}})^T A^{\mathfrak{f}}(q_\lambda^{\mathfrak{f}} + q_p^{\mathfrak{f}}) + (\hat{q}_\lambda^{\mathfrak{f}})^T A^{\mathfrak{f}}(q_\lambda^{\mathfrak{f}} + q_p^{\mathfrak{f}}), \tag{2.8}$$

and

$$L_{\mathcal{T}}(\hat{\Lambda}) = (\bar{q}^{\mathfrak{m}})^T \left(A^{\mathfrak{m}}\hat{q}_\lambda^{\mathfrak{m}} - B^{\mathfrak{m}}\hat{p}_\lambda^{\mathfrak{m}}\right) + (\hat{p}^{\mathfrak{f}})^T \left(f^{\mathfrak{f}} - (B^{\mathfrak{m},\mathfrak{f}})^T\bar{q}^{\mathfrak{m}} - (B^{\mathfrak{f}})^T\bar{q}^{\mathfrak{f}}\right) + (\bar{q}^{\mathfrak{f}})^T A^{\mathfrak{f}}\hat{q}_\lambda^{\mathfrak{f}}. \tag{2.9}$$

By grouping the terms, the definition is equivalent to the shorter form:

$$a_{\mathcal{T}}(\tilde{\Lambda},\hat{\Lambda}) = (\hat{q}^{\mathfrak{m}})^T A^{\mathfrak{m}} q^{\mathfrak{m}} + (\hat{q}^{\mathfrak{f}})^T A^{\mathfrak{f}} q^{\mathfrak{f}}, \tag{2.10}$$

where $q^{\mathfrak{m}} = q_\lambda^{\mathfrak{m}} + q_p^{\mathfrak{m}}$ and similarly for the other terms. And if we recall the definition of the matrices $A^{\mathfrak{m}}$ and $A^{\mathfrak{f}}$ as integrals, we go back to the discrete functions to see that

$$\begin{aligned} a_{\mathcal{T}}(\Lambda,\hat{\Lambda}) &= a(u_h(p_h,\lambda_h), u_h(\hat{p}_h,\hat{\lambda}_h)) \\ &= \int_{\Omega^{\mathfrak{m}}} (\mathcal{K}^{\mathfrak{m}})^{-1} u_h^{\mathfrak{m}}(p_h,\lambda_h)\cdot u_h^{\mathfrak{m}}(\hat{p}_h,\hat{\lambda}_h) + \int_{\Omega^{\mathfrak{f}}} (\mathcal{K}^{\mathfrak{f}})^{-1} u_h^{\mathfrak{f}}(p_h,\lambda_h)\cdot u_h^{\mathfrak{f}}(\hat{p}_h,\hat{\lambda}_h). \end{aligned} \tag{2.11}$$

Thus, the discrete bilinear form (2.11) is written as an integral.

In other terms, in order to be able to apply the lemma mentioned in remark 2.3, our discrete bilinear form (2.8) should verify:

$$\sum_{j=1}^{N} a_{\Omega_j}(u_h(p_h,\lambda_h), u_h(p_h,\lambda_h)) \leq k_1\ a(u_h(p_h,\lambda_h), u_h(p_h,\lambda_h)), \tag{2.12}$$

where k_1 is the constant defined in Theorem 2.1 and:

$$\begin{aligned} &a_{\Omega_j}(u_h(p_h,\lambda_h), u_h(\hat{p}_h,\hat{\lambda}_h)) \\ &= \int_{\Omega_j^{\mathfrak{m}}} (\mathcal{K}^{\mathfrak{m}})^{-1} u_h^{\mathfrak{m}}(p_h,\lambda_h)\cdot u_h^{\mathfrak{m}}(\hat{p}_h,\hat{\lambda}_h) + \int_{\Omega_j^{\mathfrak{f}}} (\mathcal{K}^{\mathfrak{f}})^{-1} u_h^{\mathfrak{f}}(p_h,\lambda_h)\cdot u_h^{\mathfrak{f}}(\hat{p}_h,\hat{\lambda}_h). \end{aligned} \tag{2.13}$$

In our case:

$$a_{\Omega_j}(u_h(p_h,\lambda_h), u_h(p_h,\lambda_h)) = \int_{\Omega_j^{\mathfrak{m}}} (\mathcal{K}^{\mathfrak{m}})^{-1} \|u_h^{\mathfrak{m}}(p_h,\lambda_h)\|_{L^2}^2 + \int_{\Omega_j^{\mathfrak{f}}} (\mathcal{K}^{\mathfrak{f}})^{-1} \|u_h^{\mathfrak{f}}(p_h,\lambda_h)\|_{L^2}^2,$$

and

$$a(u_h(p_h,\lambda_h), u_h(p_h,\lambda_h)) = \int_{\Omega^{\mathfrak{m}}} (\mathcal{K}^{\mathfrak{m}})^{-1} \|u_h^{\mathfrak{m}}(p_h,\lambda_h)\|_{L^2}^2 + \int_{\Omega^{\mathfrak{f}}} (\mathcal{K}^{\mathfrak{f}})^{-1} \|u_h^{\mathfrak{f}}(p_h,\lambda_h)\|_{L^2}^2.$$

We know (2.13) exists since all our mappings are local. This lets us split the discrete bilinear form (in its integral definition (2.11)) as a sum over elements, with the contributions to each term coming only from data that is local to the element. Furthermore, since at most k_1 subdomains have a non-empty intersection, $\sum_{j=1}^{N} a_{\Omega_j}(u_h(p_h,\lambda_h), u_h(p_h,\lambda_h))$ cannot exceed k_1 times $a(u_h(p_h,\lambda_h), u_h(p_h,\lambda_h))$.

In order to conclude this section, we have verified that the single-phase flow problem is compliant with the theoretical framework of "HPDDM Neumann".

2.2.2 Partitioning meshes from Discrete Fracture Matrix

For HPDDM GenEO, we need a partition with overlap of the domain.

First, let us build a partition without overlap of the Discrete Fracture Matrix domain into N subdomains, $\Omega = \cup_{i=1,\dots,N}\Omega_i$. Each subdomain is defined as $\Omega_i = \Omega_i^{\mathfrak{m}} \cup \Omega_i^{\mathfrak{f}}$, $i = 1,\dots,N$, with $\Omega_i^{\mathfrak{m}}$ (resp. $\Omega_i^{\mathfrak{f}}$) the subdomain i corresponding to the porous medium (resp. fractured medium).

Then, we build the overlap for each subdomain Ω_i, denoted $\Omega_i^\delta = \Omega_i^{\mathfrak{m},\delta} \cup \Omega_i^{\mathfrak{f},\delta}$, $i = 1, ..., N$, with $\Omega_i^{\mathfrak{m},\delta}$ (resp. $\Omega_i^{\mathfrak{f},\delta}$) the subdomain $\Omega_i^{\mathfrak{m}}$ (resp. $\Omega_i^{\mathfrak{f}}$) with overlap.

To do so, Algorithm 2 proposes a partitioning of the Discrete Fracture Matrix domain into subdomains with overlap based on the meshes $\mathcal{T}_h$ and $\mathcal{K}_h$.

We denote $\mathcal{T}_{h,i}$ tetrahedral mesh composing the porous subdomain $\Omega_i^{\mathfrak{m}} = \cup_{T \in \mathcal{T}_{h,i}} T$. We denote $\mathcal{T}_{h,i}^\delta$ the tetrahedral mesh composing the overlap of the subdomain $\Omega_i^{\mathfrak{m}}$. Then, each subdomain with overlap $\Omega_i^{\mathfrak{m},\delta}$, $i = 1, ..., N$, writes $\Omega_i^{\mathfrak{m},\delta} = \cup_{T \in (\mathcal{T}_{h,i} \cup \mathcal{T}_{h,i}^\delta)} T$.

Similarly, we denote $\mathcal{K}_{h,i}$ triangular mesh composing the fractures subdomain $\Omega_i^{\mathfrak{f}} = \cup_{K \in \mathcal{K}_{h,i}} K$. We denote $\mathcal{K}_{h,i}^\delta$ the triangular mesh composing the overlap of the subdomain $\Omega_i^{\mathfrak{f}}$. Then, each subdomain with overlap $\Omega_i^{\mathfrak{f},\delta}$ writes $\Omega_i^{\mathfrak{f},\delta} = \cup_{K \in (\mathcal{K}_{h,i} \cup \mathcal{K}_{h,i}^\delta)} K$.

Figure 2 displays an example of a fractured porous medium decomposed in 4 subdomains.

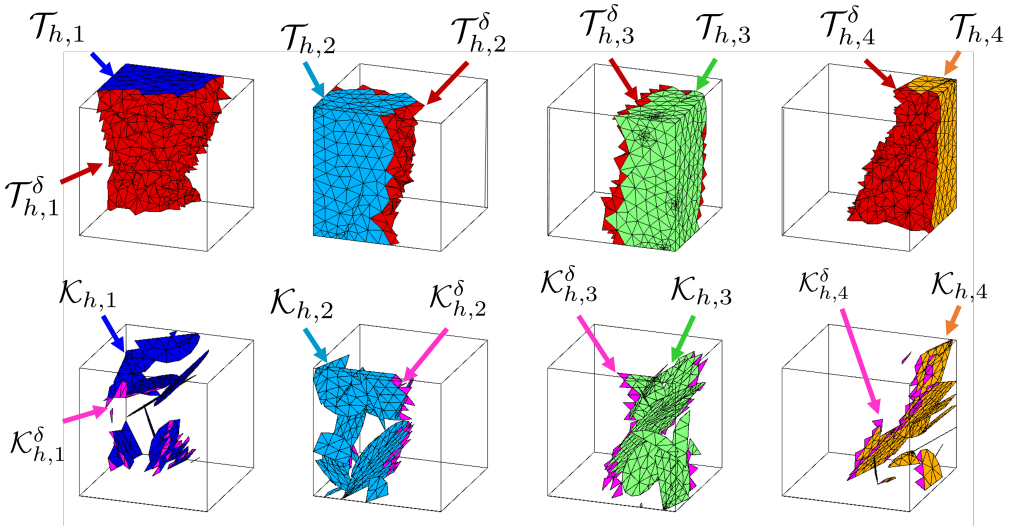


Figure 2: Domain decomposition in 4 subdomains of a fractured porous medium containing 55 fractures. The tetrahedral elements in the overlap $\mathcal{T}_{h,i}^\delta$ of the porous mesh are colored in red, the triangles in the overlap $\mathcal{K}_{h,i}^\delta$ of the fractures mesh in pink. We associated each subdomain i, $i = 1, \dots 4$, with a given color for clarity.

Algorithm 2 Partitioning of the Discrete Fracture Matrix domain based on the mesh.

1: The tetrahedral sub-meshes $\mathcal{T}_{h,i}$, $i \in [\![1, N]\!]$, which define the subdomains $\Omega_i^{\mathfrak{m}}$, are obtained by partitioning the volume mesh $\mathcal{T}_h$ with METIS [46].
2: **for** $i \leftarrow 1$ to N **do**
3: The tetrahedral elements in $\mathcal{T}_{h,i}^\delta$ are the neighbors, not in $\mathcal{T}_{h,i}$, of the inner tetrahedral elements in $\mathcal{T}_{h,i}$, which means they share a face with elements in $\mathcal{T}_{h,i}$ but do not belong to $\mathcal{T}_{h,i}$. The elements in $\mathcal{T}_{h,i}^\delta$ form one layer of overlap of the inner sub-mesh $\mathcal{T}_{h,i}$.
4: The triangular sub-meshes $\mathcal{K}_{h,i}$, which define the subdomains $\Omega_i^{\mathfrak{f}}$, are defined by all the fracture faces in $\mathcal{T}_{h,i}$.
5: The triangles in the overlap $\mathcal{K}_{h,i}^\delta$ are defined by all the fracture faces in $\mathcal{T}_{h,i}^\delta$. We also add all the neighbors, not in $\mathcal{K}_{h,i}$ and not already in $\mathcal{K}_{h,i}^\delta$, of the inner triangle elements in $\mathcal{K}_{h,i}$, which means they share an edge with elements in $\mathcal{K}_{h,i}$ but do not belong to $\mathcal{K}_{h,i}$. The elements in $\mathcal{K}_{h,i}^\delta$ form one layer of overlap of the inner sub-mesh $\mathcal{K}_{h,i}$.
6: **end for**

Now, let us explain in more details the different stages of Algorithm 2. We perform a loop over

the different subdomains i, $i = 1, \dots N$.

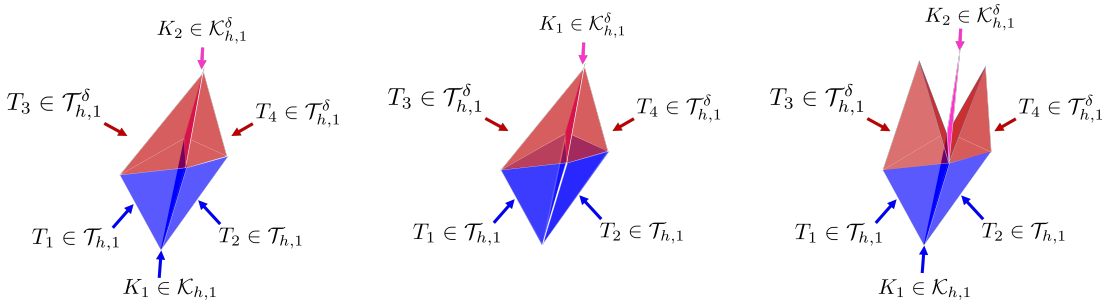
In Stage 1, to partition the Discrete Fracture Matrix, let us consider the undirected graph whose nodes are the tetrahedral elements, with an edge between two nodes if the corresponding tetrahedral elements share a face, and none otherwise. From such a graph, one can associate an adjacency matrix, which is symmetric of size $\mathcal{N}(\mathcal{T}_h) \times \mathcal{N}(\mathcal{T}_h)$. The elements of the matrix are 1 if there is an edge between two nodes, and 0 otherwise.

Then we use a mesh partitioner like METIS [46] or Scotch [58] to perform the domain decomposition. Here, for convenience, we use the MEX implementation of METIS for MATLAB implementation.

To perform Stage 3, we have constructed a structure that, for each face, provides the two tetrahedral elements that share it. From this structure, it is easy to identify the neighbors of elements in $\mathcal{T}_{h,i}$ that lie in the overlap.

Stage 4 is rather straightforward as it simply collects the fracture faces of elements in $\mathcal{T}_{h,i}$.

Stage 5 finds the triangles in the overlap in two steps. These two steps are illustrated on Figure 3 with the notation $\mathcal{F}(\mathcal{T}_h) = \{F \in \mathcal{F}_h, F \text{ is a face of } T, T \in \mathcal{T}_h\}$. First, it collects the fractures faces in the overlap $\mathcal{T}^\delta_{h,i}$ (Figure 3a). Notice there might be triangles belonging to fractures that end in the overlap (Figure 3b). Second, it collects the triangles that cannot be reached in $\mathcal{T}^\delta_{h,i}$ but can be reached through the boundary edges of $\mathcal{K}_{h,i}$ (Figure 3c).



(a) $K_2 \in \mathcal{F}(\mathcal{T}^\delta_{h,1})$ with a neighbor $K_1 \in \mathcal{K}_{h,1}$

(b) $K_1 \in \mathcal{F}(\mathcal{T}^\delta_{h,1})$ with no neighbor in $\mathcal{K}_{h,1}$

(c) $K_2 \notin \mathcal{F}(\mathcal{T}^\delta_{h,1})$

Figure 3: Sketch of a mesh to display the different configurations of triangles in the overlap $\mathcal{K}^\delta_{h,1}$.

2.2.3 Generating Neumann matrices in a simple example without fractures

HPDDM GenEO requires the following matrix structures:

1. local Neumann matrices A^δ_i;
2. a map from global unknowns to unknowns local to each subdomain, which is essentially are data structures encoding R_i and $\widetilde{R}_i$;
3. the global matrix A.

The Neumann matrix A^δ_i of the subdomain $\Omega_i, i \in [\![1, N]\!]$, is obtained by assembling the problem on the local subdomain Ω^δ_i (with one layer of overlap). As mentioned in section 2.1, it corresponds to a discretization of the problem endowed with natural boundary conditions. In order to illustrate the different matrices involved in Algorithm 1, let us consider a simple 2D case of a porous domain $\Omega = (0, 2)^2$ meshed with 8 triangles, as displayed on Figure 4. We only consider a single-phase flow in a 2D rock matrix domain without fractures and considering permeameter boundary conditions [49] in the direction x.

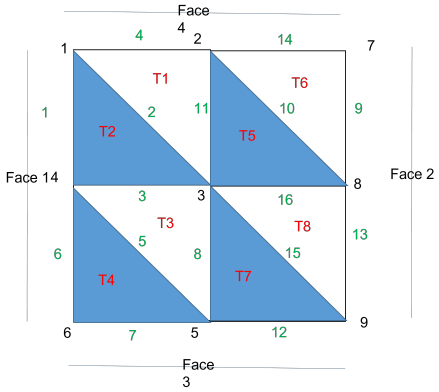


Figure 4: Sketch a porous domain $\Omega = (0,2)^2$ meshed with 8 triangles. Edges 1, 6, 9 and 13 are Dirichlet edges and edges 4, 14, 7 and 12 are homogeneous Neumann edges.

Domain decomposition.

As the domain does not have fractures, we perform stages 1, 2 and 3 of Algorithm 2 to obtain the subdomains Ω_i, $i = 1, \ldots, N$, adapted to triangular elements instead of tetrahedral. Thus, we skip Stages 4 and 5. For this example, let us consider $N = 2$ subdomains.

Figure 5 displays the domain decomposition in $N = 2$ subdomains of the porous domain Ω displayed on Figure 4.

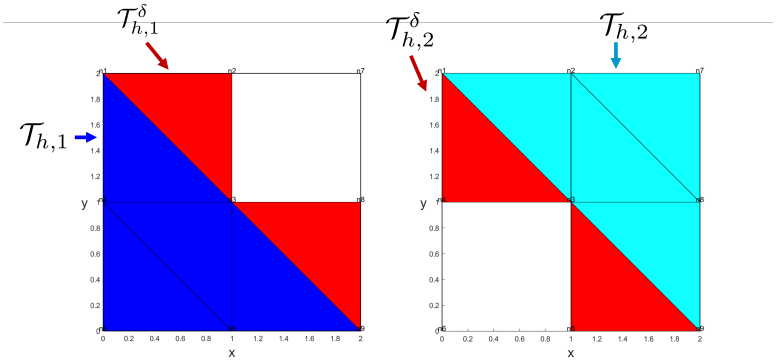


Figure 5: Domain decomposition in 2 subdomains. The triangles in the overlap $\mathcal{T}_{h,i}^{\delta}$, $i \in \{1,2\}$, are colored in red. We associate each subdomain Ω_i to a given color for clarity.

Degrees of freedom and numbering in the linear system.

The degrees of freedom (dofs) of the problem are the edges of the domain that are not Dirichlet, i.e. edges whose indices are $\mathcal{I}^{\text{edges}} = \{2, 3, 4, 5, 7, 8, 10, 11, 12, 14, 15, 16\}$. The number of dofs is $\mathcal{N}(\mathcal{I}^{\text{edges}}) = 12$.

The numbering of the dofs in the linear system $\mathcal{I}^{\text{dofs}} = \{1, \ldots, 12\}$ is displayed on Figure 6.

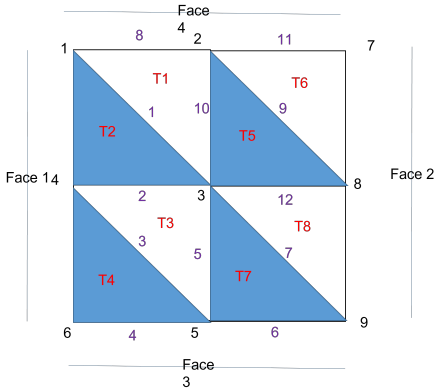


Figure 6: Numbering of the dofs in the linear system, displayed in violet.

Algebraic partition of unity

In our example, based on the decomposition described above, we have:

- the set of inner dofs indices (no overlap) $\mathcal{I}_1^{\text{dofs}} = \{1, 2, 3, 4, 5, 6, 7\}$ and $\mathcal{I}_1^{\text{dofs}} = \{8, 9, 10, 11, 12\}$;
- the set of dofs indices with overlap $\mathcal{I}_1^{\text{dofs},\delta=1} = \{1, 2, 3, 4, 5, 6, 7, 8, 10, 12\}$ and $\mathcal{I}_2^{\text{dofs},\delta=1} = \{1, 2, 5, 6, 7, 8, 9, 10, 11, 12\}$;

We consider the set of indices $\mathcal{I}_i^{\text{dofs},\delta=1}$, $i = \{1, 2\}$ to define the partition of unity. Therefore the matrix R_1 is of size $\mathcal{N}(\mathcal{I}_1^{\text{dofs},\delta=1}) \times \mathcal{N}(\mathcal{I}^{\text{dofs}}) = 10 \times 12$ and R_2 is of size $\mathcal{N}(\mathcal{I}_2^{\text{dofs},\delta=1}) \times \mathcal{N}(\mathcal{I}^{\text{dofs}}) = 10 \times 12$. Therefore the matrices R_1 and R_2 write:

$$R_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, R_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

We define Boolean partition of unity, as in [44]. We define the matrix $\widetilde{D}_1$ by setting at 1 all inner dofs that are in $\mathcal{I}_1^{\text{dofs}}$ and 0 to the dofs in the overlap. Similarly, We define the matrix $\widetilde{D}_2$ by setting at 1 all inner dofs that are in $\mathcal{I}_2^{\text{dofs}}$ and 0 to the dofs in the overlap. We obtain:

$$\widetilde{D}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \widetilde{D}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Using the notation from [44], we define the matrices $\widetilde{R}_i = \widetilde{D}_i R_i$, $i \in \{1, 2\}$. Note that the matrices $\{\widetilde{R}_i\}_{i=1}^N$ are the same operators as $\{R_i\}_{i=1}^N$ except that coefficients on the overlap are set to 0. One checks easily that $\{\widetilde{R}_i R_i^T\}_{i=1}^N$ also defines a partition of unity. This is the partition used in the PETSc implementation of HPDDM [44].

Neumann matrices

The Neumann matrices A_i^δ of size $\mathcal{N}(\mathcal{I}_i^{\text{dofs},\delta=1})$ are obtained using an assembly procedure over the subdomain mesh $\mathcal{T}_{h,i}^\delta$, $i = \{1, 2\}$. In our particular example, they write:

$$A_1^\delta = \begin{pmatrix} 8 & -2 & 0 & 0 & 0 & 0 & 0 & -2 & -2 & 0 \\ -2 & 4 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 8 & -2 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 4 & 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & -2 & 8 & 0 & 0 & -2 \\ -2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 2 \end{pmatrix},$$

$$A_2^\delta = \begin{pmatrix} 8 & -2 & 0 & 0 & 0 & -2 & 0 & -2 & 0 & 0 \\ -2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & -2 & 8 & 0 & 0 & 0 & 0 & -2 \\ -2 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8 & -2 & -2 & -2 \\ -2 & 0 & 0 & 0 & 0 & 0 & -2 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & -2 & 0 & 0 & 4 \end{pmatrix}.$$

Local matrices

The matrix A of the linear system, of size $\mathcal{N}(\mathcal{I}^{\text{dofs}})$ is obtained using an assembly procedure over the mesh $\mathcal{T}_h$. In our example, $\mathcal{N}(\mathcal{I}^{\text{dofs}}) = 12$, and A writes

$$A = \begin{pmatrix} 8 & -2 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -2 & 0 & 0 \\ -2 & 4 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 8 & -2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 4 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & -2 & 8 & 0 & 0 & 0 & 0 & -2 \\ -2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & -2 & -2 & -2 \\ -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -2 & 0 & 0 & 4 \end{pmatrix}.$$

In practice, the full matrix A can be deduced from the inner dofs of the Neumann matrices, thus providing the Neumann matrices is sufficient to obtain A.

We recall that A_i is the local matrix of size $\mathcal{N}(\mathcal{I}^{\text{dofs},\delta=1})$ defined by: $A_i = R_i A R_i^T$. In our

example:

$$A_1 = \begin{pmatrix} 8 & -2 & 0 & 0 & 0 & 0 & 0 & -2 & -2 & 0 \\ -2 & 4 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 8 & -2 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 4 & 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & -2 & 8 & 0 & 0 & -2 \\ -2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 4 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} 8 & -2 & 0 & 0 & 0 & -2 & 0 & -2 & 0 & 0 \\ -2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & -2 & 8 & 0 & 0 & 0 & 0 & -2 \\ -2 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8 & -2 & -2 & -2 \\ -2 & 0 & 0 & 0 & 0 & 0 & -2 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & -2 & 0 & 0 & 4 \end{pmatrix}.$$

Local generalized eigenvalue problem (GEVP)

To define the local GEVP, we use the partition of unity $\{\widetilde{R_i}R_i^T\}_{i=1}^N$ defined above. In our example:

$$\widetilde{R_1}R_1^T A_1 R_1 \widetilde{R_1}^T = \begin{pmatrix} 8 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & 4 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 8 & -2 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 4 & 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & -2 & 8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\widetilde{R_2}R_2^T A_2 R_2 \widetilde{R_2}^T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8 & -2 & -2 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 4 \end{pmatrix}.$$

In PETSc/HPPDM implementation, SLEPc solves the GEVP on each subdomain Ω_i concurrently.

3 Hardware and software implementation

3.1 Hardware

Most simulations presented from now are run on the supercomputer Joliot-Curie located at CEA's Very Large Computing Center (TGCC), AMD Irene ROME, hereafter referred to as [ROME], 2292 dual-processor AMD Epyc Rome computer nodes at 2.6 GHz with 64 cores per processor, for a total of 293,376 computing cores and a power of 11.75 Pflop/s, 256 GB DDR4 memory/node.

A few simulations are run on the local Inria cluster, CLEPS, on a partition hereafter referred to as [HOMO] with 2x Cascade Lake Intel Xeon 5218, 16 cores, 2.4GHz, 192 GB, 2,667 MHz.

We do not use hyperthreading in our simulations.

3.2 Software implementation

The mesh of the fractured porous medium, modeled as a Discrete Fracture Matrix, is performed in two subsequent stages thoroughly described in [49].

All the 2D conforming surface meshes are generated with `MODFRAC` [1] [16, 50]. All the 3D volume conforming meshes are generated with `GHS3D` [2] [37]. The mesh software are run on the computational resources presented in [49].

We take the same considerations mesh qualities $\delta(K)$ for 2D triangles and $\delta(T)$ for 3D tetrahedra from [49]. The quality threshold for `MODFRAC` is chosen equal to 10^{-4}, and we do not impose a quality threshold for `GHS3D`. Also, the user can choose a gradation parameter, and in this work it is equal to 2.0.

Flow simulations presented in this work are performed with different software.

First, the assembly phase, and the eventual construction of Neumann matrices, are performed with the MATLAB code `nef-flow-fpm`, which implements the mixed-hybrid finite element method, described in [48]. In particular, what is new with respect to [49] is that `nef-flow-fpm` is able to partition DFMs and generate Neumann matrices following the methods described in Sections 2.2.2 and 2.2.3 respectively. We recall that `nef-flow-fpm` is able to store the linear system in PETSc [9, 8] binary format.

Second, the the linear system (1.11) is solved with PETSc built-in solvers, totally outside `nef-flow-fpm`.

Third, the solution obtained with PETSc is post-processed with `nef-flow-fpm`.

With respect to the studied domain decomposition preconditioners, their implementation and integration to PETSc is thouroughly explained in [44].

The GenEO-based preconditioner "HPDDM GenEO" launcher, hereafter called `neumann.c`, is a modified version of `ex76.c`, which is a PETSc KSP tutorial. It takes as arguments the left-hand side matrix A, the right-hand side vector b, the Neumann matrices, their sizes their local-to-global numbering and the number of subdomains.

The splitting-based preconditioner "HPDDM split." launcher, hereafter called `splitting.c`, is a modified version of [40]. It takes as arguments A, b and the number of subdomains. The rest of the work, in particular the construction of the coarse space, is done by the HPDDM backend. There is a ParMETIS call in order to partition A of the linear system. Additional work is done in order to apply this partitioning to b.

The one-level domain decomposition preconditioners ASM and RAS launcher, hereafter called `asm_ras.c`, is also a modified version of [40], very similar to `splitting.c` and with the same entries and functionalities.

[1] `MODFRAC` is a proprietary software owned by Inria project-teams GAMMA3 and SERENA and the University of Technology of Troyes.

[2] `GHS3D` is a proprietary software owned by Inria project-team GAMMA3.

Due to the large number of simulations to run, a parser launcher [51] is used in [ROME] environment. It generates the input parameter files for the simulations launched with `neumann.c`, `splitting.c` or `asm_ras.c`.

4 Very large test case generation

4.1 Global test case generation

As a reminder, the chosen modelling strategy of the network of fractures is the Discrete Fracture Network (DFN) approach. Fractures are modelled as ellipses or disks whose properties (orientation, size, position, transmissivity) are governed by statistical laws derived from field observations [22, 14, 28, 27]. In this work, the DFNs are generated with the software DFN.lab. By considering the surrounding rock matrix, we obtain a Discrete Fracture Matrix model.

In this work, we follow the construction of test cases found in [49], where test cases up to 90,000 fracture have been generated.

4.1.1 Geometries with large and very large number of fractures

In this work, we generate very large fractured porous media ("FPM") test cases up to 697k fractures. Hereafter, the geometries are named with respect to the cube size. With respect to the test case naming in [49], we drop the "FPM" particle, since all simulations presented in this work are exclusively in fractured porous media.

Table 1: DFM "FPM" geometrical properties.

Geometry name	$N^{\mathbb{f}}$	#intersections
[L60]	39,456	66,181
[L80]	90,447	155,362
[L100]	174,315	305,455
[L130]	377,269	674,705
[L160]	696,839	1,260,231

#intersections stands for the number of fracture-fracture intersections.

As stated, only geometries with a large number of fractures, less than 100k, such as [L60] and [L80], have been studied in the literature, in particular in [49]. However, geometries with a very large number of fractures, more than 100k, such as [L100], [L130] and [L160] have not been studied in the literature in the DFM (2D+3D) framework.

4.1.2 Meshes

We add to the geometry names from Section 4.1.1 either "geo" or "fine". "geo" means that the mesh step is automatically set by `MODFRAC` according to the geometry. "fine" means that we manually chose a finer mesh step.

The quality threshold imposed to `MODFRAC`, equal to $1.00 \cdot 10^{-4}$ and is respected for all the meshes. No quality threshold imposed to `GHS3D`. Table 2 gives the mesh properties of the presented geometries in this work with `MODFRAC` and `GHS3D`. Tables 3 and 4 give the mesh quality for the above examples with `MODFRAC` and `GHS3D` respectively.

Table 2: Mesh properties for very large test cases

Mesh name	$\mathcal{N}(\mathcal{T}_h)$	$\mathcal{N}(\mathcal{F}_h)$	$\mathcal{N}(\mathcal{K}_h)$	$\mathcal{N}(\mathcal{E}_h)$
[L60geo]	7,876,897	15,825,076	1,790,168	2,751,907
[L60fineA]	16,197,529	32,501,063	2,737,569	4,236,628
[L60fineB]	29,827,028	59,807,738	4,024,218	6,214,235
[L60fineC]	59,549,925	119,337,725	6,321,429	9,719,152
[L80geo]	16,544,945	33,206,750	3,825,145	5,860,352
[L100geo]	30,454,172	61,082,397	7,058,863	10,801,781
[L130geo]	59,869,820	120,004,957	14,122,726	21,603,994
[L160geo]	102,593,939	205,561,258	24,385,046	37,327,575

Table 3: Properties of the 2D surface meshes.

Mesh name	mean $\delta(K)$	$\min_K$ $\delta(K)$	$\min_K$ $\lvert K\rvert$
[L60geo]	0.85	1.05e-04	1.39e-09
[L60fineA]	0.90	1.15e-04	2.01e-08
[L60fineB]	0.93	2.62e-04	2.01e-08
[L60fineC]	0.95	1.60e-04	2.01e-08
[L80geo]	0.84	1.11e-04	3.58e-09
[L100geo]	0.84	9.01e-04	5.59e-09
[L130geo]	0.83	1.04e-04	9.45e-09
[L160geo]	0.82	1.00e-04	1.43e-08

Table 4: Properties of the 3D volume meshes.

Mesh name	mean $\delta(T)$	$\min_T$ $\delta(T)$	$\min_T$ $\lvert T\rvert$	median $\delta(T)$	$\mathcal{N}(\mathcal{T}_h)$ s.t. $\delta(T) < 0.1$	% $\mathcal{N}(\mathcal{T}_h)$ s.t. $\delta(T) < 0.1$
[L60geo]	0.62	1.69e-06	1.53e-12	0.65	88.9k	1.13%
[L60fineA]	0.71	3.54e-06	5.33e-11	0.74	83.7k	0.52%
[L60fineB]	0.75	1.07e-06	5.08e-11	0.77	85.3k	0.29%
[L60fineC]	0.76	8.88e-07	5.33e-11	0.79	99.2k	0.16%
[L80geo]	0.61	6.90e-08	1.36e-13	0.63	216k	1.30%
[L100geo]	0.61	6.41e-07	4.05e-13	0.63	447.7k	1.47%
[L130geo]	0.60	1.90e-08	5.51e-13	0.63	964.9k	1.61%
[L160geo]	0.59	2.56e-09	1.01e-12	0.62	1.89M	1.84%

Thanks to Tables 3 and 4, and similarly to [49], for the "geo" meshes, as the number of fractures increases, both meshes tend to have lower quality, as indicated by the decrease of the mean quality value $\delta(K)$ and $\delta(T)$. For the volumes meshes, we can observe this too thanks to the slight increase of the number of tetrahedral elements below a quality of 0.1. Even if the requested quality of $1.00 \cdot 10^{-4}$ is respected by the surface meshes, for geometries with a very large number of fractures, the minimum 3D quality can be as bad as $2.56 \cdot 10^{-9}$ for [L160geo]. Nevertheless, the number of low quality elements in the 3D mesh ($\delta(T) < 0.1$) still remains relatively small, less than 2% of the total number of tetrahedra. As a matter of fact, in this work we show that these meshes perform correctly for flow simulations.

4.1.3 Hydrogeological setting

For this work, we consider the same hydrogeological settings as in [49].

We recall that we use permeameter boundary conditions (BCs), both in the rock matrix and in the fracture network. Furthermore, zero source terms, both in the rock matrix and the fracture network are considered.

We also recall that we have two settings for the hydraulic conductivity in the rock matrix and the transmissivity in the fractures:

- "1v100", where the hydraulic conductivity in the rock matrix is chosen equal to $\mathcal{K}_\alpha^{\mathbb{m}} = I_3\,\mathrm{m/s}$, $\forall\alpha$, and where the fracture tangential transmissivity is chosen equal to $\mathcal{K}_\ell^{\mathbb{f}} = 100 I_2\,\mathrm{m}^2/\mathrm{s}$, $\forall\ell$ (the same for all fractures);
- "heter", where the hydraulic conductivity in the rock matrix is chosen equal to $\mathcal{K}_\alpha^{\mathbb{m}} = 10^{-8} I_3\,\mathrm{m/s}$, $\forall\alpha$, and where the fracture tangential transmissivity $\mathcal{K}_\ell^{\mathbb{f}}$ takes one value per fracture ℓ in the range $[10^{-6}, 20]\ \mathrm{m}^2/\mathrm{s}$.

We denote the maximum ratio of heterogeneity with $\mathcal{K}_{\mathrm{ratio}}^{\mathbb{m},\mathbb{f}}$. For the "1v100" setting, it is equal to 100. For the "heter" setting, even if the exact value depends on the chosen geometry, it is always close to 10^7.

4.1.4 Global linear systems

Thanks to our code `nef-flow-fpm`, we are able to assemble the following linear systems, corresponding each one to a "test case" by combining a mesh from 4.1.2 and a permeability/transmissivity setting from 4.1.3. They can be found in Table 5.

Table 5: Properties of the very large linear systems: number of dofs, number of non-zero elements and the permeability/transmissivity ratio.

Test case	# dofs	nnz	$\mathcal{K}_{\mathrm{ratio}}^{\mathbb{m},\mathbb{f}}$
[L60geo-1v100]	18,525,914	134,224,135	1.00e+02
[L60geo-heter]	18,525,914	134,224,135	∼1.00e+07
[L60fineA-1v100]	36,658,783	263,406,425	1.00e+02
[L60fineB-1v100]	65,907,706	471,436,936	1.00e+02
[L60fineC-1v100]	128,881,985	918,288,659	1.00e+02
[L80geo-1v100]	38,986,205	282,941,795	1.00e+02
[L80geo-heter]	38,986,205	282,941,795	∼1.00e+07
[L100geo-1v100]	71,758,072	521,157,724	1.00e+02
[L100geo-heter]	71,758,072	521,157,724	∼1.00e+07
[L130geo-1v100]	141,421,137	1,028,204,671	1.00e+02
[L130geo-heter]	141,421,137	1,028,204,671	∼1.00e+07
[L160geo-1v100]	242,616,910	1,764,733,010	1.00e+02
[L160geo-heter]	242,616,910	1,764,733,010	∼1.00e+07

We would like to note that such linear systems are very ill-conditioned, as shown in [49] thanks to the computation of the estimation of the condition numbers. In order to give an idea of the condition numbers we work with, we recall from [49] that [L60geo-1v100] has an estimated condition number equal to $5.35 \cdot 10^{12}$. We also recall that [L20geo-heter] (not presented in this work), with a strong ratio of heterogeneity $\mathcal{K}_{\mathrm{ratio}}^{\mathbb{m},\mathbb{f}}$, has an estimated condition number equal to $6.86 \cdot 10^{16}$.

To conclude this section, thanks to specialized software like `MODFRAC` and `GHS3D`, and thanks to `nef-flow-fpm`, this work proves the ability to assemble 2D-3D single-phase flow fractured porous media test cases with a very large number of fractures. For instance, the two test cases created from the [L160] geometry with 697k fractures are the largest linear systems with 243M dofs. As seen in [49], such linear systems are very ill-conditioned, in particular under the "heter" permeability/transmissivity configuration, for instance, with condition numbers potentially around 10^{16} for the "heter" test cases.

4.2 Domain decomposition test cases

The novelty of this work is studying domain decomposition preconditioners for fractured porous media with a very large number of fractures, up to 697k. This is why we need to define specific test cases in order to assess their robustness.

4.2.1 Weak scaling

First, one can study weak scaling. This consists on fixing the DFM geometry, for instance [L60], and have multiple refinements, for instance, [L60geo], [L60fineA], [L60fineB] and [L60fineC]. The point is to ask METIS in `nef-flow-fpm` to create a number of sub-domains ("# sbdm") in order to maintain a constant number of dofs per sub-domain constant. For the weak scaling study, we only choose "1v100" permeability/transmissivity setting. Table 6 recalls the specificities of [L60geo-1v100], [L60fineA-1v100], [L60fineB-1v100] and [L60fineC-1v100] and gives their domain decomposition characteristics.

Table 6: Properties of weak scaling test cases: [L60geo-1v100], [L60fineA-1v100], [L60fineB-1v100] and [L60fineC-1v100].

Test case	# dofs	#sbdm	#dofs / sbdm
[L60geo-1v100]	18,525,914	522	35,490
[L60fineA-1v100]	36,658,783	1,033	35,488
[L60fineB-1v100]	65,907,706	1,857	35,491
[L60fineC-1v100]	128,881,985	3,631	35,495

In this work, weak scaling studies are performed with the test cases presented in 6, with a number of dofs per sub-domain close to 35.5k, where the "geo" mesh is the coarsest mesh and the "fineC" mesh is the most refined one. The number of dofs goes from 19M to 129M and the number of subdomains goes from 522 to 3631. This is almost 7 times more between "geo" and "fineC" for both dofs and number of subdomains.

4.2.2 Strong scaling

Second, one can study the strong scaling of domain decomposition preconditioners. This consists on fixing a test case, for instance [L60geo-1v100], with 19M dofs. What varies here is the number of subdomains created by METIS in `nef-flow-fpm`. The number of dofs per subdomain is not constant number for strong scaling studies. We choose the following number of subdomains: $[2^k]_{k\in[\![6,11]\!]}$.Table 7 recalls the specificities of [L60geo-1v100] and gives the domain decomposition characteristics for each subdomain.

Table 7: Properties of the strong scaling test cases, where [L60geo-1v100], with 18,525,914 dofs, is the fixed global test case, partitioned in $[2^k]_{k\in[\![6,11]\!]}$ subdomains.

#sbdm	#dofs / sbdm
64	289,467
128	144,734
256	72,366
512	36,183
1024	18,092
2048	9,046

4.2.3 Geometry complexity impact

Third, one can study the impact of the complexity of the "FPM" geometry, similarly to what has been done in [49]. In order to assess the robustness of domain decomposition preconditioners, the number of subdomains is chosen so it remains constant for each test case. Thus, the studied test cases are both "1v100" and "heter" versions of [L60geo], [L80geo], [L100geo], [L130geo] and [L160geo], respectively with 522, 1098, 2022, 3985 and 6825 subdomains. Table 8 recalls the specificities of the mentioned test cases and gives the domain decomposition characteristics.

Table 8: Properties of the very large test cases: number of fractures, number of dofs, number of subdomains, number of dofs per subdomain and the permeability/transmissivity ratio.

Test case	$N^{\mathbb{f}}$	# dofs	# sbdm	#dofs / sbdm	$\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$
[L60geo-1v100]	39k	18,525,914	522	35,490	1.00e+02
[L60geo-heter]	39k	18,525,914	522	35,490	∼1.00e+07
[L80geo-1v100]	90k	38,986,205	1098	35,507	1.00e+02
[L80geo-heter]	90k	38,986,205	1098	35,507	∼1.00e+07
[L100geo-1v100]	174k	71,758,072	2022	35,489	1.00e+02
[L100geo-heter]	174k	71,758,072	2022	35,489	∼1.00e+07
[L130geo-1v100]	377k	141,421,137	3985	35,488	1.00e+02
[L130geo-heter]	377k	141,421,137	3985	35,488	∼1.00e+07
[L160geo-1v100]	697k	242,616,910	6825	35,548	1.00e+02
[L160geo-heter]	697k	242,616,910	6825	35,548	∼1.00e+07

Table 8 shows the ability of `nef-flow-fpm` to assemble 2D-3D single-phase flow fractured porous media Neumann matrices from test cases with a very large number of fractures, even under the "heter" permeability/transmissivity configuration. For instance, [L160geo-heter] with 697k fractures and the highest $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$ needs the largest number of subdomains in order to maintain a constant number of dofs per subdomain with respect to the other test cases, around 35.5k.

5 Numerics with large and very large fracture networks

In this section we present many numerical experiments for flow in fractured porous media. Since the number of test cases is high and since each test case has a considerable number of unique properties, in each section we recall systematically their main properties for each one. If the reader wants the exact detail of these properties, they can refer to Section 4.

5.1 Performance of solvers and preconditioners in large-scale simulations

In order to obtain some first results with different different iterative solvers and different domain decomposition preconditioners, we choose [L60geo-1v100] and [L60geo-heter] test cases. Their detailed description can be found in Section 4, but we remind in Table 9 their main characteristics, as well as their domain decomposition characteristics.

Table 9: Main properties of test cases [L60geo-1v100] and [L60geo-heter]

Test case	$N^{\mathbb{f}}$	# dofs	# sbdm	$\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$
[L60geo-1v100]	39k	18,525,914	522	1.00e+02
[L60geo-heter]	39k	18,525,914	522	∼1.00e+07

This choice of test cases lets us to have a large test case to start with, with 39k fractures and 18.5M dofs, and also have the heterogeneity from the "heter" configuration, which makes the linear system much more ill-conditioned. See [49] for more details on the subject.

5.1.1 Conjugate Gradient (CG)

Since the linear system is square, sparse, symmetric, positive and definite, the first tested solver is naturally the Conjugate Gradient (CG). Its tolerance is set to $1.00 \cdot 10^{-14}$, and this is valid for the rest of this work. This choice tolerance is thoroughly explained in [49]. The tolerance value is very low in order to ensure flux continuity as much as possible, which is a recurrent local mass conservation issue for single-phase flow simulations in fractured porous media with a large number of fractures.

For CG, we assess the following parallel preconditioners:

- Additive Schwarz Method (ASM); a list of parameters parsed to `asm_ras.c` (see Section 3.2 for details) can be found in Appendix A.2.2;
- Two-level domain decomposition method with a GenEO-based coarse space "HPDDM GenEO"; a list of parameters parsed to `neumann.c` (see Section 3.2 for details) can be found in Appendix A.2.3;
- Two-level domain decomposition method with a splitting-based coarse space "HPDDM split."; a list of parameters parsed to `splitting.c` (see Section 3.2 for details) can be found in Appendix A.2.4.

The number of MPI processes parsed to the launchers is equal to the number of subdomains mentioned in Table 9. For the remainder of this section, this number of set to 522. The maximum number of iterations for CG is set to 10,000. As a reminder, since CG is a symmetrical solver, the only compatible preconditioners are the symmetric ones, and the only possible side for the preconditioning is the left.

We also note that both "HPDDM" methods need a number of processors for their direct solver. We choose them to be equal to 4. Furthermore, while the calculation of the number of deflation vectors is automatic in HPDDM GenEO, on the other hand, HPDDM split. requires that number as an input. That is why a good approximation of the necessary number for HPDDM split. is the maximum number of deflation vectors computed by HPDDM GenEO.

Table 10 presents the exact number of iterations and the clock time for the mentioned simulations with CG.

Table 10: Time and iterations of CG preconditioned with ASM, HPDMM split. and HPDDM GenEO.

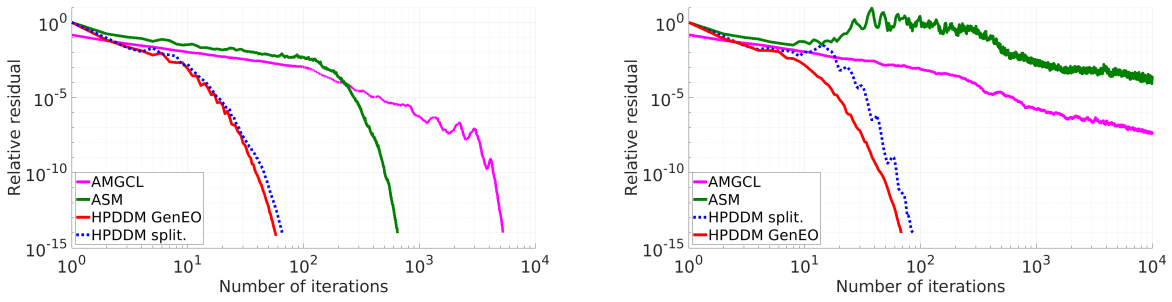
Test case	Precond.	# it.	Main stage mm:ss	PCSetUp mm:ss	KSPSolve mm:ss
[L60geo-heter]	ASM	10,000 <!>	05:48	00:02	05:23
[L60geo-1v100]	ASM	648	00:50	00:02	00:22
[L60geo-heter]	HPDDM split.	86	01:35	00:50	00:31
[L60geo-1v100]	HPDDM split.	65	01:25	00:46	00:23
[L60geo-heter]	HPDDM GenEO	67	00:59	00:35	00:17
[L60geo-1v100]	HPDDM GenEO	57	00:52	00:33	00:13

The symbol "<!>" means that the simulation stopped at the maximum number of iterations, equal to 10k, not converging down to the requested tolerance value of 10^{-14}. This only happened with [L60geo-heter]. For ASM, the minimum residual is $8.35 \cdot 10^{-5}$ obtained at iteration n^o 9898.

We have three clock times that come from the PETSc log file. "Main stage" corresponds to the total time, "PCSetUp" corresponds to the preconditioner setup "KSPSolve" corresponds to the Krylov resolution. Hereafter, we indicate these times for every simulation.

We note that for [L60geo-1v100], HPDDM GenEO computed 107 deflation vectors at most for a given subdomain, and that for [L60geo-heter], HPDDM GenEO computed 111 deflation vectors at most for a given subdomain. The quantities were used as inputs for HPDDM split. for their respective test cases.

Figure 7 presents the relative residual versus the number of iterations of CG, truncated at 10,000 iterations. We also display the results of the algebraic multi-grid preconditioner AMGCL obtained from [49].



(a) "1v100". (b) "heter".

Figure 7: CG convergence with different preconditioners on test cases [L60geo-1v100] (left) and [L60geo-heter] (right) - relative residual (log-scale) versus the number of iterations (log-scale), for the following preconditioners: AMGCL [49], ASM, HPDDM split. and HPDDM GenEO.

We recall that, in the case of PCG-AMGCL, for [L60geo-1v100] CG converged in 5,249 iterations, and that for [L60geo-heter] CG did not converge in 10k iterations down to 10^{-14}. If we only consider the first 10k iterations, the minimum residual is $4.02 \cdot 10^{-8}$ obtained at iteration 9802.

With respect to CG performance with different preconditioners, we can draw conclusions from Table 10 and Figure 7.

In the "1v100" case, the AMG preconditioner presents more than 3k iterations, while the domain decomposition methods present at most 648 iterations, which corresponds to the one-level ASM preconditioner. Even better, both two-level domain decomposition preconditioners, HPDDM split. and HPDDM GenEO, present a similar number of iterations, 57 and 65 respectively. Going

from more than 3k iterations down to around 60 iterations is a huge improvement with respect to the number of iterations. This demonstrates that domain decomposition preconditioners, despite some work on the partitioning, should be the preferred method in order to help the linear solvers to converge with a low number of iterations, at least for low $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$ test cases.

Speaking of transmissivity, the "heter" case makes almost all preconditioners to diverge, except for HPDDM split. and HPDDM GenEO, which not only converges in 86 iterations (resp. 67) and within a very reasonable main stage time, with only 10 more seconds (resp. 7 more seconds) than the "1v100" case. The single-level domain decomposition method ASM performs worse than the AMG method, even presenting very strong oscillations starting from the 10th iteration.

We remark that this is the first time that a linear solver, in this case CG, is able to converge down to 10^{-14} for an "heter" configuration with a large number of fractures in fractured porous media, and in this section being [L60geo-heter] with 39k fractures. Furthermore, this was obtained with a relatively very low number of iterations. These results demonstrate that two-level domain decomposition methods are good candidates in order to precondition a such large and ill-conditioned linear system.

5.1.2 Generalized Minimal RESidual (GMRES)

One might want to explore other Krylov methods besides CG, maybe expecting better clock times and/or number of iterations. That is why we test the performance of the iterative solver based on the Generalized Minimal RESidual Method (GMRES) [60], which is a non-symmetrical linear solver. Its tolerance is also set to 10^{-14}. The maximum number of iterations for GMRES is also set to 10,000. The restart number for GMRES is set to 90 for the remainder of this work.

For GMRES, we assess almost the same preconditioners as for CG in Section 5.1.1. However, since GMRES is non-symmetrical, it can be preconditioned either on the left or the right, and we choose the latter. Also, any preconditioner parsed to GMRES can be non-symmetrical, opening more possibilities. This is why we study the performance of the following preconditioners:

- Additive Schwarz Method (ASM): same as CG, please refer to Section 5.1.1;
- Restrictive Additive Schwarz Method (RAS): a non-symmetrical one-level domain decomposition preconditioner;
- HPDDM GenEO; almost the same as CG, but in this case, one can choose the Schwarz method for the fine level:
  - HPDDM GenEO - ASM, symmetrical; which was the actual choice for CG;
  - HPDDM GenEO - RAS, non-symmetrical;

  a list of parameters parsed to `neumann.c` (see 3.2 for details) can be found in Appendix A.2.3;
- HPDDM split.; almost the same as CG, but similarly to HPDDM GenEO, we also have two possibilities:
  - HPDDM split. - ASM, symmetrical; which was the actual choice for CG;
  - HPDDM split. - RAS, non-symmetrical;

  a list of parameters parsed to `splitting.c` (see 3.2 for details) can be found in Appendix A.2.4.

Table 11 presents the exact number of iterations and the clock time for the mentioned simulations with GMRES.

Table 11: Time and iterations of GMRES preconditioned with ASM, RAS, HPDMM split. (ASM and RAS) and HPDDM GenEO (ASM and RAS).

Test case	Precond.	# it.	Main stage mm:ss	PCSetUP mm:ss	KSPSolve mm:ss
[L60geo-heter]	ASM	10,000 <!>	07:13	00:02	06:47
[L60geo-1v100]	ASM	1396	01:28	00:02	01:00
[L60geo-heter]	RAS	1,080	01:13	00:02	00:47 @@
[L60geo-1v100]	RAS	924	01:07	00:02	00:41
[L60geo-heter]	HPDDM split. - ASM	69	01:24	00:48	00:20
[L60geo-1v100]	HPDDM split. - ASM	65	01:21	00:46	00:19
[L60geo-heter]	HPDDM split. - RAS	52	01:21	00:48	00:17
[L60geo-1v100]	HPDDM split. - RAS	72	01:23	00:47	00:20
[L60geo-heter]	HPDDM GenEO - ASM	60	00:51	00:35	00:10
[L60geo-1v100]	HPDDM GenEO - ASM	58	00:47	00:33	00:09
[L60geo-heter]	HPDDM GenEO - RAS	39	00:47	00:35	00:06
[L60geo-1v100]	HPDDM GenEO - RAS	38	00:44	00:33	00:06

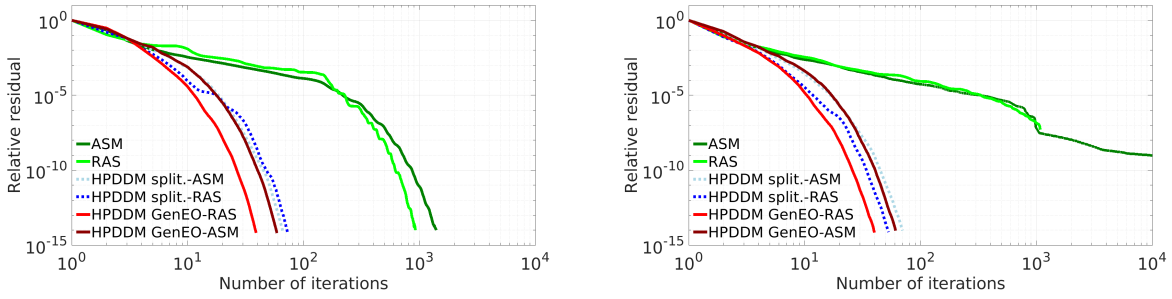
We recall that GMRES relative residuals are decreasing with respect to the number of iterations, making the last iteration the one with the smallest relative residual value.

The symbol "@@" means that GMRES suffered from a diverged breakdown. This happened with the test case [L60geo-heter] with the RAS preconditioner. Even if 1k+ iterations were performed, the iterative solver stopped at iteration n^o 1080, with relative residual equal to $5.04 \cdot 10^{-8}$, thus not converging down to the requested tolerance value of 10^{-14}.

The symbol "<!>" means that the simulation stopped at the maximum number of iterations, equal to 10k, not converging down to 10^{-14}. This happened with ASM for test case [L60geo-heter] with a minimum residual equal to $9.85 \cdot 10^{-10}$.

We recall that the linear systems being the same between the CG and GMRES resolutions, for [L60geo-1v100], HPDDM GenEO computed 107 deflation vectors at most for a given subdomain, and for [L60geo-heter], HPDDM GenEO computed 111 deflation vectors at most for a given subdomain. The quantities were used as inputs for HPDDM split. for their respective test cases.

Figure 8 presents the relative residual versus the number of iterations of GMRES, truncated at 10,000 iterations.



(a) "1v100".

(b) "heter". RAS stops at 1078 iterations due to GMRES breakdown.

Figure 8: GMRES convergence with preconditioners ASM, RAS, HPDDM split. - RAS, HPDDM split. - ASM, and HPDDM GenEO - RAS and HPDDM GenEO - ASM, on test cases [L60geo-1v100] (left) and [L60geo-heter] (right) - relative residual (log-scale) versus the number of iterations (log-scale).

With respect to GMRES performance with different preconditioners, we can draw conclusions from Table 11 and Figure 8.

In the "1v100" configuration, where $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$ is relatively low, the single-level domain decomposition methods, ASM and RAS, converge, with around 1k iterations. Nonetheless, in the "heter" configuration, RAS makes GMRES suffer from a breakdown and ASM makes the solver to not converge in 10k iterations. We conclude that such preconditioners are not compatible with single-phase flow simulations in fractured porous media with a large number of fractures and with a strong $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$ contrast. That is the reason why we do not continue their study in this work.

In both [L60geo-1v100] and [L60geo-heter] test cases, both HPDDM preconditioners converge and perform well with respect to the number of iterations and main stage time, staying in the same order of magnitude of the CG results.

In the case of HPDDM split., one has to note the effect of the choice between ASM and RAS for the fine-level. For instance, even if HPDDM split. - ASM performs better than HPDDM split. - RAS in the "1v100" configuration, with a strong $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$ in "heter", HPDDM split. - RAS gains 25% iterations with respect to HPDDM split. - ASM.

In the case of HPDDM GenEO, the effect is more notorious both in "1v100" and "heter" configurations, gaining respectively 34% and 35% with HPDDM GenEO - RAS with respect to HPDDM GenEO - ASM. Furthermore, main stage times are slightly better with HPDDM GenEO - RAS rather than with HPDDM GenEO - ASM.

That is why, in this work, when using GMRES preconditioned by some HPDDM-based method, we choose a RAS method for the fine level, despite the theory described in 2.1 only being valid for ASM. We observe in practice that the number of iterations is lower with RAS.

Also, if we compare both HPDDM split. - RAS, hereafter simply named as "HPDDM split.", and HPDDM GenEO - RAS, hereafter simply named as "HPDDM GenEO", the HPDDM GenEO main stage times are almost twice as better as HPDDM split.. In terms of iterations, there is also some significant gain, at least 25%.

With respect to the simulations with CG, at least for the [L60geo-heter] case, GMRES presents more stability than CG. Also, then number of iterations is much lower with GMRES than CG, especially with HPDDM GenEO. That is why for the rest of this work, GMRES is the chosen iterative solver to run the simulations.

In order to conclude that the two-level domain decomposition preconditioners HPDDM split. and HPDDM GenEO, despite some work on the partitioning, especially for the latter with Neumann matrices construction, are the preferred methods in this work in order to help the chosen linear solver GMRES to converge with a low number of iterations and total clock-time, for any $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$ configuration. For now, only a large number of fractures geometry, [L60] with 39k fractures, has been studied. We look forward to study very large geometries, up to 697k fractures, in the following sections.

5.2 Two-level domain decomposition preconditioners

In this section, we study the performances of GMRES preconditioned with the splitting-based coarse space two-level domain decomposition preconditioner "HPDDM split.", and the GenEO-based coarse space two-level domain decomposition preconditioner "HPDDM GenEO", whose theoretical frameworks have been presented in section 2.1 respectively. We do so by looking at three different characteristics a linear solver should have: weak scalability, strong scalability and, for the fractured porous media single-phase flow problem, robustness with respect to geometrical complexity of the DFM.

The parameters for GMRES and for the two mentioned preconditioners are chosen the same as in section 5.1.2. In particular, for GMRES, the tolerance is set to $1.00 \cdot 10^{-14}$, the maximum number of iterations is equal to 10,000 and the restart is equal to 90.

In particular, HPDDM split. requires a number of deflation vectors as input. As mentioned in section 5.1, we first simulate the test case with HPDDM GenEO, which computes automatically the number of the deflation vectors, and this number is then parsed to HPDDM split..

Also, these two-level preconditioners require a number of MPI processes required for the factorization of the coarse level, hereafter named `coarse_p`. In each section we also detail this parameter.

5.2.1 Weak scaling with a large number of fractures

We assess the weak scaling of HPDDM GenEO and HPDDM split. with the large test cases [L60geo-1v100], [L60fineA-1v100], [L60fineB-1v100] and [L60fineC-1v100] presented in Section 4. We recall their main characteristics in Table 12.

Table 12: Properties of weak scaling test cases: [L60geo-1v100], [L60fineA-1v100], [L60fineB-1v100] and [L60fineC-1v100].

Test case	$N^{\mathbb{f}}$	$\mathcal{N}(\mathcal{T}_h)$	# dofs	#sbdm	$\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$
[L60geo-1v100]	39k	7.9M	18.5M	522	10^2
[L60fineA-1v100]	39k	16M	36.7M	1,033	10^2
[L60fineB-1v100]	39k	30M	65.9M	1,857	10^2
[L60fineC-1v100]	39k	60M	129M	3,631	10^2

We recall that the number of dofs per subdomain is constant equal to roughly 35.5k.

For both preconditioners, [L60geo-1v100], [L60fineA-1v100], [L60fineB-1v100] and [L60fineC-1v100] are called with the following `coarse_p` values respectively: 4, 8, 12 and 24.

We can find in Table 13 the results of the simulations, in particular, the clock time and the number of iterations.

Table 13: Weak scaling - Time and iterations of GMRES preconditioned with HPDDM GenEO and HPDDM split.

Test case	# MPI procs.	Precond. HPDDM	# it.	Main stage mm:ss	PCSetUp mm:ss	KSPSolve mm:ss
[L60geo-1v100]	522	split.	72	01:23	00:47	00:20
[L60fineA-1v100]	1033	split.	69	01:59	00:56	00:31
[L60fineB-1v100]	1857	split.	69	02:53	01:18	00:50
[L60fineC-1v100]	3631	split.	¡¡!!	¡¡!!	¡¡!!	¡¡!!
[L60geo-1v100]	522	GenEO	38	00:44	00:33	00:06
[L60fineA-1v100]	1033	GenEO	39	01:02	00:42	00:08
[L60fineB-1v100]	1857	GenEO	39	01:18	00:40	00:10
[L60fineC-1v100]	3631	GenEO	39	01:49	01:03	00:14

The "¡¡!!" symbol means that the simulation did not run due to an out of memory error, thus no performance can be indicated. This only happened once with HPDDM split. at test case [L60fineC-1v00].

Figure 9 presents the relative residual versus the number of iterations of GMRES for this weak scalability study.

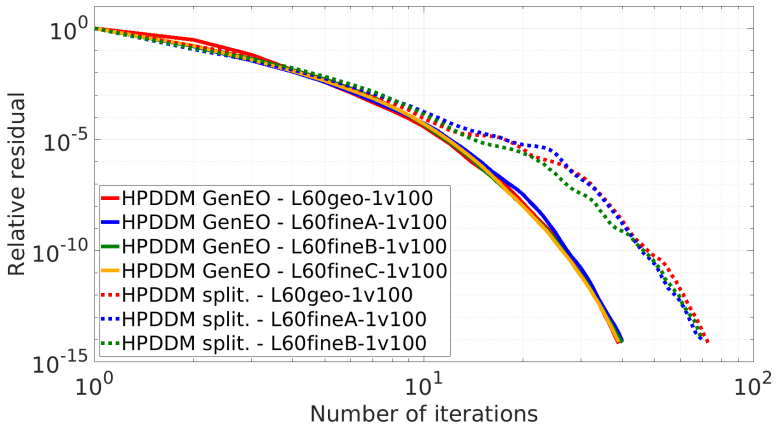


Figure 9: Study of GMRES weak scaling with preconditioners HPDDM split. (dotted lines) and HPDDM GenEO (plain lines) - relative residual (log-scale) versus the number of iterations (log-scale).

Even if [L60fineC-1v100] could not be run with HPDDM split. probably due to its size, most probably the problem being the algebraic partition done with ParMETIS which is consumes a lot of computational resources, we still have some interesting results from this weak scaling study.

As announced in Section 5.1.2, the two-level domain decomposition preconditioners converge both in less than 72 iterations, according to Figure 9, which, by the way, is less than GMRES restart set to 90. Thus, there is no effect of the restart in these simulations. For all test cases, according to Table 13, HPDDM split. stays between 69 and 72 iterations. HPDDM GenEO stays almost exactly at 39 iterations, except for [L60geo-1v100] with 38 iterations. Even though HPDDM GenEO shows a better convergence rate, both preconditioners show they are weakly scalable in terms of iterations. In fact, even if the problem size grows for the same geometry due to refinement, the number of iterations remains stable for both methods.

Nonetheless, we note that the solver clock time does not scale that well for both methods. On the one hand, for HPDDM split, the main stage time more than doubles between [L60geo-1v100] and [L60fineB-1v100], according to Table 13. HPDDM GenEO does slightly better than HPDDM split with the same test cases, with 1.75 times more main stage time. This phenomenon can be explained thanks to Table 14.

Table 14: Coarse-space and eigenvalue properties for weak scaling test cases.

Test case	Precond. HPDDM	max # nev	min # nev	median # nev	avg # nev	std dev # nev	Coarse space size
[L60geo-1v100]	split.	121	29	83	69	15	36.3k
[L60fineA-1v100]	split.	104	33	68	72	12	74.3k
[L60fineB-1v100]	split	103	30	86	69	11	128k
[L60geo-1v100]	GenEO	107	25	90	69	15	36.0k
[L60fineA-1v100]	GenEO	100	32	83	68	11	70.1k
[L60fineB-1v100]	GenEO	102	27	76	63	10	116k
[L60fineC-1v100]	GenEO	89	26	62	57	9	207k

We observe in Table 14 that the coarse space size for the test case [L60fineB-1v100] is 3.53 times (resp. 3.22) larger than the one from [L60geo-1v100] for HPDDM split (resp. HPDDM GenEO). The sudden growth of the coarse space is an issue for the direct solver used in the factorization step of the coarse space construction in both preconditioners.

For HPDDM GenEO we observe that, even if the mean number of computed eigenvalues stays

quite stable, between 63 and 69 per subdomain, the refinements between [L60geo-1v100] and [L60fineB-1v100] make the standard deviation to drop from 15 to 10. This means that the number of eigenvalues computed by the preconditioner is much better distributed between subdomains for each refinement. Despite this, the maximum number of computed eigenvalues stays above 100. The only refinement where this value drops below 100 is with [L60fineC-1v100], but at the cost of a 1.78 times larger coarse space with respect to [L60fineB-1v100].

The same analysis could be performed for HPDDM split. Nonetheless, HPDDM split. shows a worse coarse space construction if we take into account the statistics presented in 14, the number of iterations and the clock times shown in 13. Furthermore, [L60fineC-1v100] could not be performed with this preconditioner, showing that, at first glance, HPDDM split. does not perform well with very large test cases with more than 100M dofs.

In order to conclude this section, both HPDDM GenEO and HPDDM split. are weakly scalable with respect to the number of iterations. However with larger test cases, in terms of dofs, the construction of the coarse space can become costly, thus affecting the total clock time. This is especially true for HPDDM split. preconditioner, not even being capable of solving the largest test case with finest mesh yielding 129M dofs.

5.2.2 Strong scaling with a large number of fractures

We assess the strong scaling of HPDDM GenEO and HPDDM split. with the large test case [L60geo-1v100] presented in Section 4. We recall that the [L60] geometry has 39k fractures and the global linear system from [L60geo-1v100] has 7.9M tetrahedra and 18.5M dofs. The main properties of the strong scaling test cases, named after the number of MPI processes, are recalled in Table 15.

Table 15: Properties of the strong scaling test cases generated from [L60geo-1v100], with 39k fractures and 7.9M tetrahedra and with 18.5M dofs in its global linear system, partitioned in $[2^k]_{k\in[\![6,11]\!]}$ subdomains.

#sbdm	#dofs / sbdm
64	289k
128	144k
256	72.4k
512	36.2k
1024	18.1k
2048	9.05k

We recall that, even if the number of dofs per subdomain is not constant, we solve exactly the same linear system for each number of MPI processes. In the case of HPDDM GenEO, this supposes a different set of Neumann matrices parsed to `neumann.c` for each number of MPI processes, even if the left hand side and right hand side of the linear system do not change between each simulation.

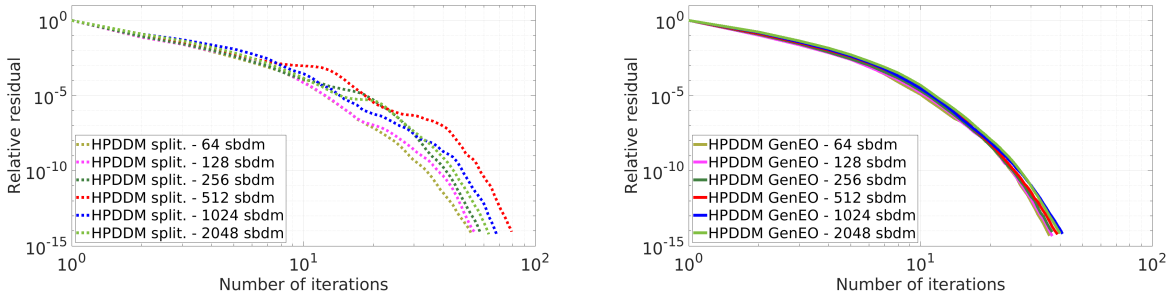
For both preconditioners, `coarse_p` takes the following values: 2, 2, 3 4, 6 and 7, respectively for 64, 128, 256, 512, 1024 and 2048 MPI processes.

We can find in Table 16 the results of the simulations, in particular, the clock time and the number of iterations.

Table 16: Strong scaling - Time and iterations of GMRES preconditioned with HPDDM GenEO and HPDDM split with [L60geo-1v100] test case.

# sbdm = # MPI procs.	Precond. HPDDM	# it.	Main stage mm:ss	PCSetUp mm:ss	KSPSolve mm:ss
64	split.	52	39:31	38:08	01:15
128	split.	54	11:18	10:31	00:35
256	split.	57	02:51	02:17	00:21
512	split.	78	01:26	00:50	00:18
1024	split.	67	00:52	00:21	00:14
2048	split.	62	00:49	00:15	00:14
64	GenEO	35	19:36	18:47	00:40
128	GenEO	36	05:02	04:37	00:17
256	GenEO	36	01:39	01:25	00:08
512	GenEO	38	00:43	00:32	00:05
1024	GenEO	40	00:28	00:16	00:05
2048	GenEO	39	00:25	00:13	00:05

Figure 10 presents the relative residual versus the number of iterations of GMRES for this strong scalability study.



(a) HPDDM split.

(b) HPDDM GenEO.

Figure 10: Study of GMRES strong scaling with preconditioners HPDDM split. (left in dotted lines) and HPDDM GenEO (right in plain lines) - relative residual (log-scale) versus the number of iterations (log-scale).

From Table 16 we read that the number of iterations is below 78, thus below the restart of GMRES.

With respect to main stage time, one can deduce a "sweet spot" from where raising the number of subdomains, thus the amount of computational resources, is not interesting anymore.

For HPDDM GenEO, Table 16 shows that, with 64 subdomains, the main stage time is around 20 minutes. By doubling the number of subdomains, reaching 128, the main stage time is around 5 minutes, thus gaining 4 times the amount of clock time with respect to 64 subdomains. Again, by doubling the number of subdomains, reaching 256, we gain 3 times more main stage time with respect to 128 subdomains, making it still worth it. By repeating this reasoning, we gain 2.3 with 512 (w.r.t. 256), 1.5 with 1024 (w.r.t. 512) and 1.2 with 2048 (w.r.t. 1024). Even if decreasing the main stage time in absolute value is good, going below gaining twice the main stage time with twice the amount of computational resources is not worth it. Moreover, we also observe that the KSPSolve time stagnates starting at 512 MPI procs with 5s. The 1024 and 2048 procs do not make this metric better, only the PCSetUp time slightly. This makes the 512 subdomain configuration the "optimal" choice, also known as the "sweet spot", which is the one having around 36.2k dofs

per subdomain. This justifies our choice of the number of dofs per subdomain equal to 35.5k in Sections 5.2.1 and 5.2.3.

With respect to the number of iterations, for HPDDM GenEO, Table 16 shows that the number of iterations remains between 35 and 40, meaning that this preconditioner respects strong scalability. This is not the case of HPDDM split., where the number of iterations varies between 52 and 78.

Furthermore, Figure 10a shows that, GMRES preconditioned with HPDDM split. presents many more oscillations than the respective simulations done with HPDDM GenEO, represented in Figure 10b. This means that, HPDDM GenEO is a far more robust preconditioner than HPDDM split. with respect to rounding errors.

With respect to the eigenvalues computed with both preconditioners, Table 17 presents some statistics.

Table 17: Coarse-space and eigenvalue properties for strong scaling test cases.

# sbdm	Precond. HPDDM	max # nev	min # nev	median # nev	avg # nev	std dev # nev	Coarse space size
64	split.	363	135	249	251	58	16.0k
128	split.	231	68	140	163	35	20.9k
256	split.	159	49	134	109	22	27.9k
512	split.	112	30	32	70	14	35.8k
1024	split.	79	15	37	45	10	46.4k
2048	split.	50	6	25	29	7.0	60.1k
64	GenEO	368	134	278	239	54	15.3k
128	GenEO	249	64	155	151	36	20.6k
256	GenEO	161	56	137	107	23	27.3k
512	GenEO	115	33	54	70	14	35.6k
1024	GenEO	79	15	49	46	10	46.9k
2048	GenEO	58	8	26	30	7.3	60.8k

We can see from Table 17 that both methods present a significant enhancement to the quality of the subdomains as the number of subdomains increases. For instance, for HPDDM split., the standard deviation goes from 58 (with 64 MPI processes) to 7 (with 2048 MPI processes), thus decreasing by more than 8 times. A low standard deviation means that the distribution of the number of computed eigenvalues is more equilibrated.

This decrease can be also observed in the other statistics, except for the size of the coarse space. The latter behaves differently by growing with the number of subdomains, going from 16k (with 64 MPI processes) to more than 60k (with 2048 MPI processes) thus growing by 3.76 times. The growth of the coarse space size might explain partially why we do not gain more clock time with a larger number of MPI processes. In fact, since the factorization of the coarse space is done with a direct solver, we know that these kind of solvers do not scale with the size of the problem. Furthermore, one does not have to forget the number of possible communications between subdomains.

To conclude this section, HPDDM GenEO is strongly scalable with respect to the number of iterations. Strong scaling results with HPDDM split. are less encouraging, as the number of iterations is far less centered than HPDDM GenEO. Furthermore, choosing around 35.5k dofs per subdomain ensures a "sweet spot" where the clock time is well-balanced with the number of MPI processes, which represents the computational resources.

5.2.3 Impact of geometries with a very large number of fractures

We assess the performance of GMRES preconditioned with HPDDM split. and HPDDM GenEO with large test cases, such as [L60geo-1v100], [L60geo-heter], [L80geo-1v100] and [L80geo-heter], with less than 100k fractures, but also with very large test cases, such as [L100geo-1v100], [L100geo-heter], [L130geo-1v100], [L130geo-heter], [L160geo-1v100] and [L160geo-heter], with more than 100k fractures, presented in Section 4. The main properties of these test cases, as well as their domain decomposition configurations, are recalled in Table 18.

Table 18: Properties of geometry impact test cases: [L60geo-1v100], [L60geo-heter], [L80geo-1v100], [L80geo-heter], [L100geo-1v100], [L100geo-heter], [L130geo-1v100], [L130geo-heter], [L160geo-1v100] and [L160geo-heter].

Test case	$N^{\mathbb{f}}$	$\mathcal{N}(\mathcal{T}_h)$	# dofs	#sbdm	# dofs / sbdm	$\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$
[L60geo-1v100]	39k	7.88M	18.5M	522	35.5k	10^2
[L80geo-1v100]	90k	16.5M	39.0M	1,098	35.5k	10^2
[L100geo-1v100]	174k	30.4M	71.8M	2,022	35.5k	10^2
[L130geo-1v100]	377k	59.9M	141M	3,985	35.5k	10^2
[L160geo-1v100]	697k	103M	243M	6,825	35.5k	10^2
[L60geo-heter]	39k	7.88M	18.5M	522	35.5k	10^7
[L80geo-heter]	90k	16.5M	39.0M	1,098	35.5k	10^7
[L100geo-heter]	174k	30.4M	71.8M	2,022	35.5k	10^7
[L130geo-heter]	377k	59.9M	141M	3,985	35.5k	10^7
[L160geo-heter]	697k	103M	243M	6,825	35.5k	10^7

For both preconditioners, `coarse_p` takes the following values: 4, 8, 16, 32 and 96 respectively for [L60], [L80], [L100], [L130] and [L160] geometry-related test cases.

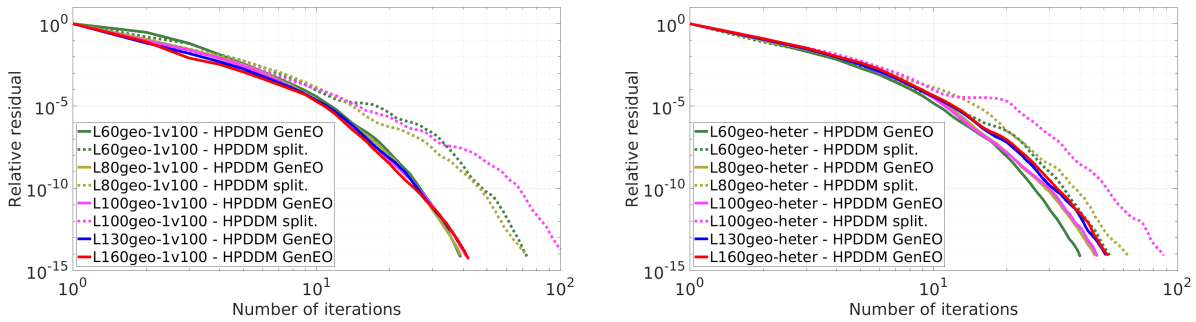
We can find in Table 19 the result of the simulations, in particular, the clock time and the number of iterations.

Table 19: Geometry impact - Time and iterations of GMRES preconditioned with HPDDM GenEO and HPDDM split for large and very large test cases.

Test case	# MPI procs.	Precond. HPDDM	# it.	Main stage mm:ss	PCSetUp mm:ss	KSPSolve mm:ss
[L60geo-1v100]	522	split.	72	01:23	00:47	00:20
[L80geo-1v100]	1098	split.	71	02:14	01:06	00:35
[L100geo-1v100]	2022	split.	103	03:10	01:24	01:02
[L130geo-1v100]	3985	split.	¡¡!!	¡¡!!	¡¡!!	¡¡!!
[L160geo-1v100]	6825	split.	¡¡!!	¡¡!!	¡¡!!	¡¡!!
[L60geo-1v100]	522	GenEO	38	00:44	00:33	00:06
[L80geo-1v100]	1098	GenEO	38	01:07	00:47	00:08
[L100geo-1v100]	2022	GenEO	40	01:41	01:09	00:13
[L130geo-1v100]	3985	GenEO	41	03:11	02:11	00:24
[L160geo-1v100]	6825	GenEO	41	04:09	02:33	00:37
[L60geo-heter]	522	split.	52	01:21	00:48	00:17
[L80geo-heter]	1098	split.	62	02:15	01:10	00:35
[L100geo-heter]	2022	split.	87	03:33	01:44	01:17
[L130geo-heter]	3985	split.	¡¡!!	¡¡!!	¡¡!!	¡¡!!
[L160geo-heter]	6825	split.	¡¡!!	¡¡!!	¡¡!!	¡¡!!
[L60geo-heter]	522	GenEO	39	00:47	00:35	00:06
[L80geo-heter]	1098	GenEO	45	01:15	00:54	00:10
[L100geo-heter]	2022	GenEO	46	01:57	01:23	00:16
[L130geo-heter]	3985	GenEO	50	03:39	02:11	00:28
[L160geo-heter]	6825	GenEO	51	05:39	03:52	00:49

The "¡¡!!" symbol means that the simulation did not run due to an out of memory error, thus no performance can be indicated. This only happened once with HPDDM split. with [L130] and [L160] geometry-related test cases.

Figure 11 presents the relative residual versus the number of iterations of GMRES for the study of the impact of the geometry on the solver convergence, truncated at 100 iterations for "1v100" configurations.



(a) "1v100". (b) "heter".

Figure 11: Study of geometry impact on GMRES convergence preconditioned with HPDDM split. and HPDDM GenEO for two configurations, "1v100" (right) and "heter" (left) - relative residual (log-scale) versus the number of iterations (log-scale).

From Table 19 we deduce that the number of iterations is below 87, except for [L100geo-1v100] with HPDDM split, where it reaches 103, being above the restart of GMRES. The results presented

in Table 19, where the presented test cases with a large and very large number of fractures converge with a relatively low number of iterations in a reasonable main stage time, are an innovation with respect to the literature, in particular [49].

With respect to the number of iterations, for both preconditioners, as the number of fractures grows, the number of iterations grows as well.

For low permeability/transmissivity contrast $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$ in "1v100" configuration, the number of iterations of HPDDM GenEO remains between 38 for [L60geo-1v100] and 41 for [L160geo-1v100], according to Table 19. This is visible in Figure 11a, where all the HPDDM GenEO curves are very close to each other. This means that this preconditioner scales very well with respect to the number of fractures, even for very large test cases up to 697k fractures and 243M dofs. This is not the case of HPDDM split., where the number of iterations varies between 72 and 102, making it non-adapted to such very large-scale simulations. We also add that the good of HPDDM GenEO non-oscillating behavior observed in Section 5.2.2 is still present here. This could not be assessed with HPDDM split. since the test cases [L130geo-1v100] and [L160geo-1v100] are out of its scope. This could be anticipated from Section 5.2.1, where the size of the linear system becomes a challenge for HPDDM split.

For large permeability/transmissivity contrast $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$ in "heter" configuration, the amplitude of the number of iterations of HPDDM GenEO becomes wider, going from 39 for [L60geo-heter] to 51 for [L160geo-heter] according to Table 19, and much more visible in Figure 11b. For [L60geo-heter], a large scale simulation with 39k fractures and a large $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$, HPDDM GenEO loses "only" one iteration with respect to its "1v100" counterpart [L60geo-1v100], also a large scale simulation with but with a low $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$. Nonetheless, for [L160geo-heter], a very large scale simulation with 697k fractures and a large $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$, HPDDM GenEO loses ten iterations with respect to its counterpart [L60geo-1v100], also a very-large scale simulation but with a low $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$. For "heter" test cases, even though the number of iterations remains acceptably close to each other, the effects of a strong $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$ contrast make convergence slower. HPDDM GenEO does not suffer that much from the scalability with respect to the number of fractures, but rather from adding some permeability/transmissivity contrast.

With respect to clock time, for both preconditioners, by comparing to [49] there is a huge improvement, going from hours to only minutes thanks to the parallelism. We can even make converge very complex test cases, such as the "heter" ones with a large and a very large number of fractures, down to 10^{-14} in a reasonable time, which was impossible to do with AMG preconditioners.

Even though HPDDM split. cannot be studied fully due to the RAM issues, we can deduct that, independently from the $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$, the growing number of fractures has an impact on the main stage time. In fact, thanks to Table 19 we observe that between [L60geo-1v100] and [L100geo-1v100], and between [L60geo-heter] and [L100geo-heter] the main stage time more than doubles.

Thanks to HPDDM GenEO we can study the real very large scale challenge, which is the explosion of the main stage time. Even if we keep a similar number of dofs per subdomain, the main stage time is multiplied by 5.66 between [L60geo-1v100] and [L160geo-1v100]. The effect is worsened if we add a strong $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$ contrast, where the main stage time is multiplied by more than 7 times between [L60geo-heter] and [L160geo-heter]. Between [L60geo-1v100] and [L60geo-heter] there is a 2s difference, while between [L160geo-1v100] and [L160geo-heter] there is a 1min30s difference. Even if HPDDM GenEO is able to simulate very large test cases, the underperformance with respect to main stage time is due to the presence of 697k fractures and 243M dofs. The effect is amplified by the presence of a strong $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$ contrast.

A study of the distribution of the eigenvalues computed by these two-level preconditioners, presented in Table 20, might enlighten us with some explanations.

Table 20: Coarse-space and eigenvalue properties for strong scaling test cases.

Test case	Precond. HPDDM	max # nev	min # nev	median # nev	avg # nev	std dev # nev	Coarse space size
[L60geo-1v100]	split.	121	29	83	69	15	36.3k
[L80geo-1v100]	split.	121	30	81	72	14	79.5k
[L100geo-1v100]	split	119	30	77	76	14	153k
[L60geo-1v100]	GenEO	107	25	90	69	15	36.0k
[L80geo-1v100]	GenEO	123	32	72	72	14	79.2k
[L100geo-1v100]	GenEO	122	27	73	76	15	153k
[L130geo-1v100]	GenEO	128	26	89	80	14	319k
[L160geo-1v100]	GenEO	134	35	111	84	14	577k
[L60geo-heter]	split.	118	29	95	70	14	36.6k
[L80geo-heter]	split.	119	30	61	73	14	80.3k
[L100geo-heter]	split	123	33	91	77	14	155k
[L60geo-heter]	GenEO	111	26	93	70	15	36.8k
[L80geo-heter]	GenEO	125	30	70	73	14	80.5k
[L100geo-heter]	GenEO	124	27	72	77	15	155k
[L130geo-heter]	GenEO	133	29	88	82	14	325k
[L160geo-heter]	GenEO	139	33	107	86	14	587k

Thanks to Table 20, we can observe the effect of the number of fractures on the size of the coarse space. Both HPDDM GenEO and HPDDM split. present similar statistics. That is why we only focus on HPDDM GenEO, since it is the only one capable of simulating the largest test cases.

In the "1v100" configuration. We go from a size of 36k for [L60geo-1v100] to 577k for [L160geo-1v100], multiplying by 16 the size of the coarse space. Such a growth in the size of the coarse space explains the slow-down in the main stage time. In fact, as explained in Sections 5.2.1 and 5.2.2, the factorization step becomes too costly for the direct solver. Furthermore, adding a strong $\mathcal{K}_{\text{ratio}}^{\mathbb{m},\mathbb{f}}$ contrast amplifies this effect. The size of the coarse space from [L160geo-heter] is larger than the one from [L160geo-1v100] by 10k, while the size of the coarse space from [L60geo-heter] is larger than the one from [L160geo-1v100] by "only" 0.2k.

Also, the number of computed eigenvalues per subdomain is equally distributed among all test cases, with a standard deviation between 14 and 15. Nonetheless, the average, the median and the maximum number of computed eigenvalues is slightly more significant in the "heter" configuration rather than the "1v100" configuration. This reinforces the argument of the amplifier role of the heterogeneity of the contrast of permeability/transmissivity in very large scale simulations.

To conclude this section, with these two-level domain decomposition preconditioners, GMRES is capable of converging down to 10^{-14} with a low number of iterations, less than 45 for HPDDM GenEO, and reasonable clock times, less than 2 minutes, for test cases with a large number of fractures but less than 100k. However, if we consider test cases with a very large number of fractures, up to 697k with 243M dofs, HPDDM split. is not able to perform the simulations. HPDDM GenEO is able to tackle such test cases, still with a low number of iterations, less than 51. Nevertheless, the effect of the number of fractures makes the solver times to grow slightly, going up to more than 4 minutes. The heterogeneity of the contrast of permeability/transmissivity worsens this effect by making the clock time as large as 6 minutes for the largest test case. Despite this, these clock times remain reasonable, within the minute-scale, and we have to add the scalability with respect to the number of fractures that HPDDM GenEO respects quite well.

These simulations represent a milestone in the field of fractured porous media. As a matter of fact, a single-phase flow can now be simulated in a discrete fracture matrix with a very large

number of fractures, up to 697k, with 243M dofs.

Conclusion

In this work, the main objective was to solve efficiently the very ill-conditioned linear system, containing hundreds of millions of unknowns, arising from the mixed-hybrid finite element discretization of time-independent single-phase flow in discrete fracture matrix model.

We successfully meshed 2D-3D fractured porous media with large and very number of fractures, reaching an order of magnitude in the hundreds of thousands.

As the linear system matrix is sparse, square, symmetric, positive and definite, we investigated Krylov methods such as the conjugate gradient (CG), which is symmetric, and GMRES, non-symmetric. The chosen preconditioning technique, domain decomposition methods, presented multiple configurations. We studied single-level methods, such as the Additive Schwarz method (ASM) and Restrictive ASM (RAS), and two two-level methods equipped with coarse space operators. The first one, based on a splitting method and the second one, based on the Generalized Eivengalue Problem on the overlap (GenEO). The use of GenEO required a development of a partitioning method of the DFM mixed-dimensional mesh.

We considered test cases of increasing geometrical complexity. The chosen test cases are known to be difficult as they may contain a very large number of fractures, up to 697k, and may have very strong hydraulic conductivity/transmissivity contrasts, up to 7 orders of magnitude. We also considered test cases in the view of testing the scalability of domain decomposition preconditioners.

As observed in previous work [49], the condition number of the linear system matrix is very high for large-scale and high permeability constrast test cases. In this paper, after discarding poorly performing single-level methods, we have demonstrated that both two-level domain decomposition methods exhibited promising performance in terms of clock time and iteration count. Moreover, GMRES, a non-symmetric method, outperformed CG, making it the preferred choice. Both two-level methods showed weak scalability, although the splitting-based preconditioner struggled with very large linear systems. In terms of strong scalability, the GenEO-based method outperformed the splitting-based approach. When increasing the geometrical complexity of the DFMs by adding more fractures, the GenEO-based method proved nearly twice as efficient as the splitting-based method in both clock time and iterations. Even under strong hydraulic conductivity/transmissivity contrasts in the largest DFM, featuring 697k fractures, 243M dofs, and 6,825 MPI processes, the GenEO-based method maintained solid performance, completing the simulation in under six minutes with 51 iterations.

As future work, alternative partitioning algorithms could be explored in the HPDDM framework to achieve a better balance in the number of computed eigenpairs per subdomain.

Thanks to HPDDM GenEO, the solver time is now reasonable, even for very large-scale problems. Ongoing work aims to further reduce the total computational time of the full single-phase flow workflow by implementing an MPI-like parallel version of both the assembly phase and the post-processing stage. As a first step, the global linear system matrix has been recently reconstructed from the Neumann matrices, reducing the cost of global matrix assembly.

Acknowledgements

This work was granted access to the GENCI-sponsored HPC resources of TGCC@CEA under allocation AD010607519R2. The authors are grateful to Simon Legrand for his support with pruners, and to Raphaël Zanella for helping in the development of the different PETSc launchers. The authors also thank Fractory, a joint laboratory between ITASCA Consultants SAS, CNRS, and the University of Rennes, for supplying the geometries with very large number of fractures and the heterogeneous transmissivity field.

A Appendices

A.1 Complements for the mathematical setting

A.1.1 Linear system block sizes

The dimension of the matrices defined in section 1.3 are given in Table 21.

Matrix	dimension	matrix	dimension
$A^{\mathfrak{m}}$	$4\mathcal{N}(\mathcal{T}_h) \times 4\mathcal{N}(\mathcal{T}_h)$	$A^{\mathfrak{f}}$	$3\mathcal{N}(\mathcal{K}_h) \times 3\mathcal{N}(\mathcal{K}_h)$
$B^{\mathfrak{m}}$	$\mathcal{N}(\mathcal{T}_h) \times 4\mathcal{N}(\mathcal{T}_h)$	$B^{\mathfrak{f}}$	$\mathcal{N}(\mathcal{K}_h) \times 3\mathcal{N}(\mathcal{K}_h)$
$B^{\mathfrak{f},\mathfrak{m}}$	$4\mathcal{N}(\mathcal{T}_h) \times \mathcal{N}(\mathcal{K}_h)$		
$C^{\mathfrak{m}}$	$4\mathcal{N}(\mathcal{T}_h) \times \mathcal{N}(\mathcal{F}_h^{\mathrm{in},N})$	$C^{\mathfrak{f}}$	$3\mathcal{N}(\mathcal{K}_h) \times \mathcal{N}(\mathcal{E}_h \backslash \mathcal{E}_h^D)$

Table 21: Dimension of the matrices in equation (1.8)

A.1.2 Linear system block contents

All of the involved matrices have a natural element by element block structure. We detail the block structure, as well as the non-zero elements in the blocks. Note that all elements not explicitly defined are zero.

- The square matrices $A^{\mathfrak{f}}$ and $A^{\mathfrak{m}}$ are block diagonal, with the diagonal blocks given by

$$(A_K^{\mathfrak{f}})_{e,e'} = \int_K (w_{K,e}^{\mathfrak{f}})^T (\mathcal{K}^{\mathfrak{f}})_{|K}^{-1} w_{K,e'}^{\mathfrak{f}}, \quad (f, f') \in [1,3],$$

  and

$$(A_T^{\mathfrak{m}})_{ff'} = \int_T (w_{T,f}^{\mathfrak{m}})^T (\mathcal{K}^{\mathfrak{m}})_{|T}^{-1} w_{T,f'}^{\mathfrak{m}}, \quad (f, f') \in [1,4]$$

  where the functions $w_{K,e}^{\mathfrak{f}}$ (resp. $w_{T,f}^{\mathfrak{m}}$) are the Raviart-Thomas(-Nédélec) basis functions for triangles (resp. tetrahedra).

- Matrices $B^{\mathfrak{m}}$ and $B^{\mathfrak{f}}$ are also block diagonal matrices, with block sizes $\mathcal{N}(\mathcal{T}_h) \times \mathcal{N}(\mathcal{T}_h)$ and $\mathcal{N}(\mathcal{K}_h) \times \mathcal{N}(\mathcal{K}_h)$ respectively. Each diagonal block is a column vector in $\mathbb{R}^4$ (resp. in $\mathbb{R}^3$), with all vector elements being all ones. In other words

$$(B^{\mathfrak{m}} p^{\mathfrak{m}})_T = B_T^{\mathfrak{m}} p_T^{\mathfrak{m}}, \text{ resp. } (B^{\mathfrak{f}} p^{\mathfrak{f}})_K = B_K^{\mathfrak{f}} p_K^{\mathfrak{f}}.$$

- The matrices $C^{\mathfrak{m}}$ and $C^{\mathfrak{f}}$ are block column matrices of sizes $\mathcal{N}(\mathcal{T}_h) \times \mathcal{N}(\mathcal{F}_h^{\mathrm{in},N})$ and $\mathcal{N}(\mathcal{K}_h) \times \mathcal{N}(\mathcal{E}_h \backslash \mathcal{E}_h^D)$ respectively. Each block column $C_{T,F}^{\mathfrak{m}}$ (resp. $C_{K,E}^{\mathfrak{f}}$) is of size 4 (resp. 3), with

$$(C_{T,F}^{\mathfrak{m}})_f = 1 \text{ if the face with local number } f \text{ in } T \text{ has global number } F,$$
$$(C_{K,E}^{\mathfrak{f}})_e = 1 \text{ if the edge with local number } e \text{ in } K \text{ has global number } E.$$

  Note that these matrices have the effect of gathering the multipliers that belong to all the inner faces of a tetrahedron (resp. all the edges of a triangle).

- Last, the coupling matrix $B^{\mathfrak{m},\mathfrak{f}}$ is a block matrix of block size $\mathcal{N}(\mathcal{T}_h) \times \mathcal{N}(\mathcal{K}_h)$, where each block is a vector of size 4, with

$$(B_{T,K}^{\mathfrak{m},\mathfrak{f}})_f = -1 \text{ if the face with local number } f \text{ in } T$$
$$\text{corresponds to the fracture face with global number } K.$$

  In the same way as for the C matrices, $B^{\mathfrak{m},\mathfrak{f}}$ gathers the element hydraulic head $\boldsymbol{P}^{\mathfrak{f}}$ from all the fracture faces of a given tetrahedron.

To define the right hand sides, we need some further notation.
We first let, for each $T \in \mathcal{T}_h$ (resp. each $K \in \mathcal{K}_h$):

$$f_T^{\mathfrak{m}} = \int_T f^{\mathfrak{m}}, \quad (\text{resp. } f_K^{\mathfrak{f}} = \int_K f^{\mathfrak{f}}).$$

For the Dirichlet boundary condition, we define for each face $F \in \mathcal{F}_h^{\mathrm{D}}$ (resp. each edge $E \in \mathcal{E}_h^D$):

$$g_F^{\mathfrak{m}} = \int_F g^{\mathfrak{m},D} \quad (\text{resp. } g_E^{\mathfrak{f}} = \int_E g^{\mathfrak{f},D}).$$

Then for $T \in \mathcal{T}_h$ we define an element source vector $g_T^{\mathfrak{m}} \in \mathbb{R}^4$ by:

$$g_{T,f}^{\mathfrak{m}} = \begin{cases} g_F^{\mathfrak{m}} & \text{if } F \subset \Gamma^D, \\ 0 & \text{otherwise,} \end{cases} \text{where } F \text{ is the global number of face } f \text{ in } T, \quad f = 1, \ldots, 4.$$

Similarly, for $K \in \mathcal{K}_h$, we define an element source vector $g_T^{\mathfrak{f}} \in \mathbb{R}^3$ by:

$$g_{K,e}^{\mathfrak{f}} = \begin{cases} g_E^{\mathfrak{f}} & \text{if } E \subset \Sigma^D, \\ 0 & \text{otherwise,} \end{cases} \text{where } E \text{ is the global number of edge } e \text{ in } K, \quad e = 1, \ldots, 3.$$

Because of the block structures of all matrices involved in (1.8), it is possible to write the equations at the element level. We note that this enables us to recover a formulation that is identical with the formulation used in [41].

It will be convenient to define several sets that will let us identify the entities involved in each equation.

- For a given tetrahedron $T \in \mathcal{T}_h$, we let $\mathcal{F}(T) = \{F \in \mathcal{F}_h, F \text{ is a face of } T\}$;
- For a given face $F \in \mathcal{F}_h$, we let $\mathcal{T}(F) = \{T \in \mathcal{T}_h, F \text{ is a face of } T\}$;
- For a given triangle $K \in \mathcal{K}_h$, we let $\mathcal{T}(K) = \{T \in \mathcal{T}_h, K \text{ is a fracture face of } T\}$;
- For a given triangle $K \in \mathcal{K}_h$, we let $\mathcal{E}(K) = \{E \in \mathcal{E}_h, E \text{ is an edge of } K\}$;
- For a given edge $E \in \mathcal{E}_h$, we let $\mathcal{K}(E) = \{K \in \mathcal{K}_h, E \text{ is an edge of } K\}$.

Furthermore, we let $\mathcal{F}(T)^{\mathrm{frac}} = \mathcal{F}(T) \cap \mathcal{F}_h^{\mathrm{frac}}$ and $\mathcal{F}(T)^{\mathrm{in}} = \mathcal{F}(T) \cap \mathcal{F}_h^{\mathrm{in},N}$ (with $\mathcal{F}(T)^{\mathrm{frac}} \cup \mathcal{F}(T)^{\mathrm{in}} = \mathcal{F}(T)$), as well as $\mathcal{T}(F, F') = \mathcal{T}(F) \cap \mathcal{T}(F')$ (where K can be substituted by F) and $\mathcal{K}(E, E') = \mathcal{K}(E) \cap \mathcal{K}(E')$.

Note that the sets $\mathcal{T}(F)$ usually contain two elements, unless the face is on the boundary of the domain. On the other hand $\mathcal{T}(K)$ always contains two elements, as it is assumed that no fracture lies along the boundary of the cube.

In the rock matrix We write the discrete version of Darcy's law and the local mass conservation for each tetrahedron

$$A_T^{\mathfrak{m}} \boldsymbol{q}_T^{\mathfrak{m}} - (B_T^{\mathfrak{m}}) p_T^{\mathfrak{m}} - \sum_{K \in \mathcal{F}(T)^{\mathrm{frac}}} B_{TK}^{\mathfrak{m},\mathfrak{f}} p_K^{\mathfrak{f}} + \sum_{F \in \mathcal{F}(T)^{\mathrm{in}}} C_{TF}^{\mathfrak{m}} \lambda_F^{\mathfrak{m}} = g_T^{\mathfrak{m}}, \qquad \forall T \in \mathcal{T}_h. \quad \text{(A.1a)}$$

$$(B_T^{\mathfrak{m}})^T \boldsymbol{q}_T^{\mathfrak{m}} = f_T^{\mathfrak{m}}, \qquad \forall T \in \mathcal{T}_h. \quad \text{(A.1b)}$$

and the flux continuity across interior faces gives

$$\sum_{T \in \mathcal{T}(F)} (C_{TF}^{\mathfrak{m}})^T \boldsymbol{q}_T^{\mathfrak{m}} = 0, \qquad \forall F \in \mathcal{F}_h^{\mathrm{in},N}. \quad \text{(A.1c)}$$

In the fractures We write the discrete version of Darcy's law and the local mass conservation for each triangle

$$A_K^{\mathfrak{f}} \boldsymbol{q}_K^{\mathfrak{f}} - (B_K^{\mathfrak{f}}) p_K^{\mathfrak{f}} + \sum_{E \in \mathcal{E}(K)} C_{KE}^{\mathfrak{f}} \lambda_E^{\mathfrak{f}} = g_K^{\mathfrak{f}}, \qquad \forall K \in \mathcal{K}_h, \tag{A.2a}$$

$$(B_K^{\mathfrak{f}})^T \boldsymbol{q}_K^{\mathfrak{f}} + \sum_{T \in \mathcal{T}(K)} (B_{TK}^{\mathfrak{m},\mathfrak{f}})^T \boldsymbol{q}_T^{\mathfrak{m}} = f_K^{\mathfrak{f}}, \quad \forall K \in \mathcal{K}_h, \tag{A.2b}$$

and the flux continuity across fracture intersections gives

$$\sum_{K \in \mathcal{K}(E)} (C_{KE}^{\mathfrak{f}})^T \boldsymbol{q}_K^{\mathfrak{f}} = 0, \qquad \forall E \in \mathcal{E}_h \backslash \mathcal{E}_h^D. \tag{A.2c}$$

As usual for hybridized formulations, it is now possible to eliminate the flux unknowns at the element level, using equations (A.1a) and (A.2a). Note also that these two equations will be used to recover the fluxes after the global linear system (1.11) has been solved. In the more usual case of an equi-dimensional model, the matrix $\mathcal{B}$ is also block diagonal, but this is not the case here due to the coupling between the porous medium and the fractures.

We give below the system of equations obtained after the elimination, as well as the required matrix elements. Only the non-zero elements are defined.

Hydraulic head rock matrix

$$D_T^{\mathfrak{m}} p_T^{\mathfrak{m}} - \sum_{F \in \mathcal{F}(T)^{\text{in}}} R_{TF}^{\mathfrak{m},\mathfrak{m}} \lambda_F^{\mathfrak{m}} - \sum_{K \in \mathcal{F}(T)^{\text{frac}}} R_{TK}^{\mathfrak{m},\mathfrak{f}} p_K^{\mathfrak{f}} = \mathscr{F}_T^{\mathfrak{m}}, \quad \forall T \in \mathcal{T}_h; \tag{A.3}$$

with

$$\begin{aligned}
D_T^{\mathfrak{m}} &= (B_T^{\mathfrak{m}})^T (A_T^{\mathfrak{m}})^{-1} B_T^{\mathfrak{m}} &&= \sum_{(F,F') \in \mathcal{F}(T)^2} ((A_T^{\mathfrak{m}})^{-1})_{FF'}, && \forall T \in \mathcal{T}_h \\
R_{TF}^{\mathfrak{m},\mathfrak{m}} &= (B_T^{\mathfrak{m}})^T (A_T^{\mathfrak{m}})^{-1} C_{TF}^{\mathfrak{m}} &&= \sum_{F' \in \mathcal{F}(T)} ((A_T^{\mathfrak{m}})^{-1})_{FF'} && \forall T \in \mathcal{T}_h, \forall F \in \mathcal{F}_h^{\text{in},N}, \\
R_{TK}^{\mathfrak{m},\mathfrak{f}} &= -(B_T^{\mathfrak{m}})^T (A_T^{\mathfrak{m}})^{-1} B_{TK}^{\mathfrak{m},\mathfrak{f}} = \sum_{F \in \mathcal{F}(T)} ((A_T^{\mathfrak{m}})^{-1})_{K,F}, && && \forall T \in \mathcal{T}_h, \forall K \in \mathcal{K}_h,
\end{aligned}$$

and

$$\mathscr{F}_T^{\mathfrak{m}} = f_T^{\mathfrak{m}} - (B_T^{\mathfrak{m}})^T (A_T^{\mathfrak{m}})^{-1} g_T^{\mathfrak{m}}, \quad \forall T \in \mathcal{T}_h.$$

Multiplier, porous medium

$$-\sum_{T \in \mathcal{T}(F)} (R_{TF}^{\mathfrak{m},\mathfrak{m}})^T p_T^{\mathfrak{m}} + \sum_{F' \in \mathcal{F}_h^{\text{in},N}} M_{F,F'}^{\mathfrak{m},\mathfrak{m}} \lambda_{F'}^{\mathfrak{m}} + \sum_{K \in \mathcal{K}_h} M_{FK}^{\mathfrak{m},\mathfrak{f}} p_K^{\mathfrak{f}} = \mathscr{V}_F^{\mathfrak{m}}, \quad \forall F \in \mathcal{F}_h; \tag{A.4}$$

with

$$\begin{aligned}
M_{F.F'}^{\mathfrak{m},\mathfrak{m}} &= \sum_{T \in \mathcal{T}(F,F')} (C_{TF}^{\mathfrak{m}})^T (A_T^{\mathfrak{m}})^{-1} C_{TF'}^{\mathfrak{m}} &&= \sum_{T \in \mathcal{T}(F,F')} ((A_T^{\mathfrak{m}})^{-1})_{FF'} && \forall (F,F') \in (\mathcal{F}_h^{\text{in},N})^2, \\
M_{FK}^{\mathfrak{m},\mathfrak{f}} &= -\sum_{T \in \mathcal{T}(F,K)} (C_{TF}^{\mathfrak{m}})^T (A_T^{\mathfrak{m}})^{-1} B_{TK}^{\mathfrak{m},\mathfrak{f}} = \sum_{T \in \mathcal{T}(F,K)} (A_T^{\mathfrak{m}})^{-1})_{FK} && && \forall F \in \mathcal{F}_h^{\text{in},N}, \forall K \in \mathcal{K}_h.
\end{aligned}$$

and

$$\mathscr{V}_F^{\mathfrak{m}} = \sum_{T \in \mathcal{T}(F)} (C_{TF}^{\mathfrak{m}})^T (A_T^{\mathfrak{m}})^{-1} g_T^{\mathfrak{m}}, \quad \forall F \in \mathcal{F}_h^{\text{in},N}.$$

Hydraulic head fracture

$$-\sum_{T\in\mathcal{T}(K)}(R_{TK}^{\mathfrak{m},\mathfrak{f}})^T p_T^{\mathfrak{m}}+\sum_{F\in\mathcal{F}_h^{\mathrm{in},N}}(M_{FK}^{\mathfrak{m},\mathfrak{f}})^T\lambda_F^{\mathfrak{m}}+D_K^{\mathfrak{f}}p_K^{\mathfrak{f}}+\sum_{K'\in\mathcal{K}_h}M_{KK'}^{\mathfrak{f},\mathfrak{f}}p_{K'}^{\mathfrak{f}}$$
$$-\sum_{E\in\mathcal{E}(K)}R_{KE}^{\mathfrak{f}}\lambda_E^{\mathfrak{f}}=\mathscr{F}_K^{\mathfrak{f}}+\mathscr{V}_K^{\mathfrak{m},\mathfrak{f}},\quad\forall K\in\mathcal{K}_h \tag{A.5}$$

with

$$\begin{aligned}
D_K^{\mathfrak{f}}&=(B_K^{\mathfrak{f}})^T(A_K^{\mathfrak{f}})^{-1}B_K^{\mathfrak{f}} &&=\sum_{(E,E')\in\mathcal{E}(K)^2}((A_K^{\mathfrak{f}})^{-1})_{E,E'} &&\forall K\in\mathcal{K}_h,\\
M_{K,K'}^{\mathfrak{f},\mathfrak{f}}&=\sum_{T\in\mathcal{T}(K,K')}(B_{TK}^{\mathfrak{m},\mathfrak{f}})^T(A_T^{\mathfrak{m}})^{-1}B_{TK'}^{\mathfrak{m},\mathfrak{f}}=\sum_{T\in\mathcal{T}(K,K')}((A_T^{\mathfrak{m}})^{-1})_{K,K'} &&\forall(K,K')\in(\mathcal{K}_h)^2,\\
R_{KE}^{\mathfrak{f}}&=(B_K^{\mathfrak{f}})^T(A_K^{\mathfrak{f}})^{-1}C_{KE}^{\mathfrak{f}} &&=\sum_{E'\in\mathcal{E}(K)}((A_K^{\mathfrak{f}})^{-1})_{E,E'} &&\forall K\in\mathcal{K}_h,\forall E\in\mathcal{E}_h.
\end{aligned}$$

and

$$\begin{aligned}
\mathscr{F}_K^{\mathfrak{f}}&=f_K^{\mathfrak{f}}+(B_K^{\mathfrak{f}})^T(A_K^{\mathfrak{f}})^{-1}g_K^{\mathfrak{f}}, &&\forall K\in\mathcal{K}_h,\\
\mathscr{V}_K^{\mathfrak{m},\mathfrak{f}}&=-\sum_{T\in\mathcal{T}(K)}(B_{TK}^{\mathfrak{m},\mathfrak{f}})^T(A_T^{\mathfrak{m}})^{-1}g_T^{\mathfrak{m}}, &&\forall K\in\mathcal{K}_h.
\end{aligned}$$

Multiplier fracture

$$-\sum_{K\in\mathcal{K}(E)}(R_{KE}^{\mathfrak{f}})^T p_K^{\mathfrak{f}}+\sum_{E'\in\mathcal{E}_h\backslash\mathcal{E}_h^D}M_{EE'}^{\mathfrak{f}}\lambda_{E'}^{\mathfrak{f}}=\mathscr{V}_E^{\mathfrak{f}},\quad\forall E\in\mathcal{E}_h\backslash\mathcal{E}_h^D, \tag{A.6}$$

with

$$M_{E,E'}^{\mathfrak{f}}=\sum_{K\in\mathcal{K}(E,E')}(C_{KE}^{\mathfrak{f}})^T(A_K^{\mathfrak{f}})^{-1}C_{KE'}^{\mathfrak{f}}=\sum_{K\in\mathcal{K}(E,E')}((A_K^{\mathfrak{f}})^{-1})_{E,E'},\quad\forall(E,E')\in(\mathcal{E}_h)^2.$$

and

$$\mathscr{V}_E^{\mathfrak{f}}=\sum_{K\in\mathcal{K}(E)}(C_{KE}^{\mathfrak{m}})^T(A_K^{\mathfrak{f}})^{-1}g_K^{\mathfrak{f}},\quad\forall E\in\mathcal{E}_h\backslash\mathcal{E}_h^D.$$

We have the following dimensions:

$$\begin{aligned}
D^{\mathfrak{m}}&\in\mathbb{R}^{\mathcal{N}(\mathcal{T}_h)\times\mathcal{N}(\mathcal{T}_h)}, & R^{\mathfrak{m},\mathfrak{m}}&\in\mathbb{R}^{\mathcal{N}(\mathcal{T}_h)\times\mathcal{N}(\mathcal{F}_h^{\mathrm{in},N})}\\
R^{\mathfrak{m},\mathfrak{f}}&\in\mathbb{R}^{\mathcal{N}(\mathcal{T}_h)\times\mathcal{N}(\mathcal{K}_h)}, & M^{\mathfrak{m},\mathfrak{m}}&\in\mathbb{R}^{\mathcal{N}(\mathcal{F}_h^{\mathrm{in},N})\times\mathcal{N}(\mathcal{F}_h^{\mathrm{in},N})}\\
M^{\mathfrak{m},\mathfrak{f}}&\in\mathbb{R}^{\mathcal{N}(\mathcal{K}_h)\times\mathcal{N}(\mathcal{F}_h^{\mathrm{in},N})}, & D^{\mathfrak{f}}&\in\mathbb{R}^{\mathcal{N}(\mathcal{K}_h)\times\mathcal{N}(\mathcal{K}_h)}\\
M^{\mathfrak{f},\mathfrak{f}}&\in\mathbb{R}^{\mathcal{N}(\mathcal{K}_h)\times\mathcal{N}(\mathcal{K}_h)}, & R^{\mathfrak{f}}&\in\mathbb{R}^{\mathcal{N}(\mathcal{K}_h)\times\mathcal{N}(\mathcal{E}_h\backslash\mathcal{E}_h^D)}\\
M^{\mathfrak{f}}&\in\mathbb{R}^{\mathcal{N}(\mathcal{E}_h\backslash\mathcal{E}_h^D)\times\mathcal{N}(\mathcal{E}_h\backslash\mathcal{E}_h^D)}.
\end{aligned} \tag{A.7}$$

We also define the following matrices:

$$\begin{aligned}
A_{FF'}^{\mathfrak{m},\mathfrak{m}}&=M_{FF'}^{\mathfrak{m},\mathfrak{m}}-\sum_{T\in\mathcal{T}(FF')}(R_{TF}^{\mathfrak{m},\mathfrak{m}})^T(D_T^{\mathfrak{m}})^{-1}R_{TF'}^{\mathfrak{m},\mathfrak{m}},\\
A_{FK}^{\mathfrak{m},\mathfrak{f}}&=M_{FK}^{\mathfrak{m},\mathfrak{f}}-\sum_{T\in\mathcal{T}(FK)}(R_{TF}^{\mathfrak{m},\mathfrak{m}})^T(D_T^{\mathfrak{m}})^{-1}R_{TK}^{\mathfrak{m},\mathfrak{f}},\\
A_{KK'}^{\mathfrak{f},\mathfrak{f}}&=M_{K,K'}^{\mathfrak{f},\mathfrak{f}}-\sum_{T\in\mathcal{T}(K,K')}(R_{TK}^{\mathfrak{m},\mathfrak{f}})^T(D_T^{\mathfrak{m}})^{-1}R_{TK'}^{\mathfrak{m},\mathfrak{f}},\\
\tilde{\mathscr{F}}_F^{\mathfrak{m}}&=\sum_{T\in\mathcal{T}(F)}(R_{TF}^{\mathfrak{m},\mathfrak{m}})^T(D_T^{\mathfrak{m}})^{-1}\mathscr{F}_F^{\mathfrak{m}},\\
\tilde{\mathscr{F}}_K^{\mathfrak{m},\mathfrak{f}}&=\sum_{T\in\mathcal{T}(K)}(R_{TK}^{\mathfrak{m},\mathfrak{f}})^T(D_T^{\mathfrak{m}})^{-1}\mathscr{F}_K^{\mathfrak{m}},
\end{aligned} \tag{A.8}$$

The hydraulic head in the rock matrix can be recovered by solving (A.3) for each tetrahedron.

A.2 Parameters for the launchers

A.2.1 Common parameters

We parse the following parameters to every launcher:

- `-load_dir`: a path to the working directory;
- `-ksp_type`: the iterative solver type, which can be either `cg` or `gmres`;
- `-ksp_pc_side`: the preconditioning side, which is equal to `left` for CG and `right` for GMRES;
- `-ksp_max_it`: the maximum number of iterations, an integer which depends on the section we work on;
- `-ksp_gmres_restart`: only used when the Krylov method is GMRES, it is always equal to `90`;
- `-ksp_monitor`: giving a path generates a file with the preconditioned residual at each iteration;
- `-ksp_converged_reason`: giving a path generates a file with the total number of iterations and the convergence flag (converged, diverged, breakdown, etc.);
- `-ksp_view`: giving the value `ascii` and a path generates a file with some verbose of the preconditioner and the Kyrlov method;
- `-ksp_view_solution`: giving the value `binary` and a path generates a binary file with the solution of the linear system;
- `-info`: giving a path generates a verbose file for each process.

A.2.2 `asm_ras.c`

Besides the parameters presented in appendix A.2.1, we also parse the following parameters to `asm_ras.c`:

- `-pc_type`: the single-level domain decomposition preconditioner is called with the value `asm`;
- `-pc_asm_type`: in this work we have two possibilities for the single-level preconditioner, either `basic` for ASM or `restrict` for RAS;
- `-sub_pc_factor_mat_solver_type`: local problems are factored with the direct solver library `mumps`;
- `-sub_pc_type`: to be more specific, the factorization is done thanks to `cholesky`.

A.2.3 `neumann.c`

Besides the parameters presented in appendix A.2.1, we also parse the following parameters to `neumann.c`:

- `-pc_type`: the two-level domain decomposition preconditioner is called with the value `hpddm`;
- `-pc_hpddm_define_subdomains`: the value `1` makes the preconditioner to search in the working path for the global to local and subdomain size files;
- `-pc_hpddm_has_neumann`: the value `1` makes the preconditioner to search in the working path for the Neumann matrices, global to local and subdomain size files;

- `-pc_hpddm_levels_1_pc_asm_type`: similarly to appendix A.2.2, we have two possibilities for the fine-level preconditioner, either `basic` for ASM or `restrict` for RAS;
- `-pc_hpddm_levels_1_sub_pc_factor_mat_solver_type`: similarly to appendix A.2.2, local problems are factored with the direct solver library `mumps`;
- `-pc_hpddm_levels_1_sub_pc_type`: similarly to appendix A.2.2, the factorization in the fine level is done thanks to `cholesky`;
- `-pc_hpddm_levels_1_st_share_sub_ksp`: this option makes use the same factorization method as in the fine level for the GEVP (2.4);
- `-pc_hpddm_levels_1_eps_threshold`: it is the ν threshold for the GEVP mentioned in theorem 2.1, set is equal to `0.1` in this work;
- `-pc_hpddm_levels_1_eps_use_inertia`: the maximum number of computed eigenpairs in the GEVP is automatically computed by "HPDDM GenEO" backend for each subdomain Ω_i;
- `-pc_hpddm_coarse_pc_factor_mat_solver_type`: the Galerkin product in order to build the coarse operator is performed with the direct solver library `mumps`;
- `-pc_hpddm_coarse_pc_type`: the factorization in the Galerkin product is done thanks to `cholesky`;
- `-pc_hpddm_coarse_mat_filter`: this options filters out the values below a threshold in the coarse operator, which is set to `1e-14` in this work;
- `-pc_hpddm_coarse_p`: the number of MPI processes used for assembling the coarse operator;
- `-pc_hpddm_coarse_correction`: this is the choice we have to make in (2.5), between an `additive`, `deflated` or `balanced` two-level preconditioner; in this work, for CG it is balanced, while for GMRES it is deflated.

A.2.4 `splitting.c`

Besides the parameters presented in appendix A.2.1, the parameters parsed to `splitting.c` are exactly the ones from A.2.3, except for the following:

- `-pc_hpddm_has_neumann`: the value is set to `0`, so no Neumann matrix is used;
- `-pc_hpddm_levels_1_eps_use_inertia`: this option is not available for "HPDDM split.", thus is not used; instead, `-pc_hpddm_levels_1_eps_nev` is the maximum number of computed eigenpairs in the GEVP for all subdomains Ω_i and has to be manually set by the user.

Most importantly, we also add the option `-pc_hpddm_block_splitting` to activate the library related to the method presented in [40].

References

[1] C. Alboin. *Deux outils mathématiques pour modéliser l'écoulement et le transport de polluants dans un milieu poreux fracturé*. Phd thesis (in french), Paris IX Dauphine University (France), 2000. URL: `https://bu.dauphine.psl.eu/fileviewer/view.php?doc=2000PA090024`.

[2] C. Alboin, J. Jaffré, J. E. Roberts, X. Wang, and C. Serres. Domain Decomposition for Some Transmission Problems in Flow in Porous Media. In Z. Chen, R. E. Ewing, and Z.-C. Shi, editors, *Numerical Treatment of Multiphase Flows in Porous Media*, pages 22–34. Springer Berlin Heidelberg, 2000. `doi:10.1007/3-540-45467-5_2`.

[3] L. Amir, M. Kern, Z. Mghazli, and J. E. Roberts. Intersecting fractures in porous media: mathematical and numerical analysis. *Applicable Analysis*, 102(5):1312–1334, 2023. `doi:10.1080/00036811.2021.1981878`.

[4] L. Amir, M. Kern, J. E. Roberts, and V. Martin. Décomposition de domaine pour un milieu poreux fracturé : un modèle en 3D avec fractures. *Revue Africaine de Recherche en Informatique et Mathématiques Appliquées*, Volume 5, Special Issue TAM TAM'05, november 2006, 2006. doi:10.46298/arima.1851.

[5] P. Angot, F. Boyer, and F. Hubert. Asymptotic and numerical modelling of flows in fractured porous media. *ESAIM: Mathematical Modelling and Numerical Analysis*, 43(2):239–275, 2009. doi:10.1051/m2an/2008052.

[6] P. F. Antonietti, J. De Ponti, L. Formaggia, and A. Scotti. Preconditioning Techniques for the Numerical Solution of Flow in Fractured Porous Media. *Journal of Scientific Computing*, 86(1):2, 2020. doi:10.1007/s10915-020-01372-0.

[7] A. Arrarás, F. J. Gaspar, L. Portero, and C. Rodrigo. Mixed-Dimensional Geometric Multigrid Methods for Single-Phase Flow in Fractured Porous Media. *SIAM Journal on Scientific Computing*, 41(5):B1082–B1114, 2019. doi:10.1137/18M1224751.

[8] S. Balay, S. Abhyankar, M. F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, J. Faibussowitsch, W. D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M. G. Knepley, F. Kong, S. Kruger, D. A. May, L. C. McInnes, R. T. Mills, L. Mitchell, T. Munson, J. E. Roman, K. Rupp, P. Sanan, J. Sarich, B. F. Smith, H. Suh, S. Zampini, H. Zhang, H. Zhang, and J. Zhang. PETSc/TAO users manual. Technical Report ANL-21/39 - Revision 3.22, Argonne National Laboratory, 2024. doi:10.2172/2205494.

[9] S. Balay, W. D. Gropp, L. Curfman McInnes, and B. F. Smith. Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[10] I. Berre, W. M. Boon, B. Flemisch, A. Fumagalli, D. Gläser, E. Keilegavlen, A. Scotti, I. Stefansson, A. Tatomir, K. Brenner, S. Burbulla, P. Devloo, O. Duran, M. Favino, J. Hennicker, I-H. Lee, K. Lipnikov, R. Masson, K. Mosthaf, M. G. C. Nestola, C.-F. Ni, K. Nikitin, P. Schädle, D. Svyatskiy, R. Yanbarisov, and P. Zulian. Verification benchmarks for single-phase flow in three-dimensional fractured porous media. *Advances in Water Resources*, 147:103759, 2021. doi:10.1016/j.advwatres.2020.103759.

[11] I. Berre, F. Doster, and E. Keilegavlen. Flow in Fractured Porous Media: A Review of Conceptual Models and Discretization Approaches. *Transport in Porous Media*, 130(1):215–236, 2019. doi:10.1007/s11242-018-1171-6.

[12] S. Berrone, S. Pieraccini, and S. Scialò. Flow simulations in porous media with immersed intersecting fractures. *Journal of Computational Physics*, 345:768–791, 2017. doi:10.1016/j.jcp.2017.05.049.

[13] D. Boffi, F. Brezzi, and M. Fortin. *Mixed finite element methods and applications*, volume 44 of *Springer Series in Computational Mathematics*. Springer, Heidelberg, 2013. doi:10.1007/978-3-642-36519-5.

[14] E. Bonnet, O. Bour, N. E. Odling, P. Davy, I. Main, P. Cowie, and B. Berkowitz. Scaling of fracture systems in geological media. *Reviews of Geophysics*, 39(3):347–383, 2001. doi:10.1029/1999RG000074.

[15] W. M. Boon, J. M. Nordbotten, and I. Yotov. Robust discretization of flow in fractured porous media. *SIAM Journal on Numerical Analysis*, 56(4):2203–2233, 2018. doi:10.1137/17M1139102.

[16] H. Borouchaki, P. Laug, and P.-L. George. Parametric surface meshing using a combined advancing-front generalized delaunay approach. *International Journal for Numerical Methods in Engineering*, 49(1-2):233–259, 2000. doi:10.1002/1097-0207(20000910/20)49:1/2<233::AID-NME931>3.0.CO;2-G.

[17] K. Brenner, M. Groza, C. Guichard, G. Lebeau, and R. Masson. Gradient discretization of hybrid dimensional Darcy flows in fractured porous media. *Numerische Mathematik*, 134(3):569–609, 2016. doi:10.1007/s00211-015-0782-x.

[18] K. Brenner, J. Hennicker, R. Masson, and P. Samier. Gradient discretization of hybrid-dimensional Darcy flow in fractured porous media with discontinuous pressures at matrix-fracture interfaces. *IMA Journal of Numerical Analysis*, 37(3):1551–1585, 2017. doi:10.1093/imanum/drw044.

[19] K. Brenner, J. Hennicker, R. Masson, and P. Samier. Hybrid-dimensional modelling of two-phase flow through fractured porous media with enhanced matrix fracture transmission conditions. *Journal of Computational Physics*, 357:100–124, 2018. doi:10.1016/j.jcp.2017.12.003.

[20] S. C. Brenner. A multigrid algorithm for the lowest-order raviart–thomas mixed triangular finite element method. *SIAM Journal on Numerical Analysis*, 29(3):647–678, 1992. doi:10.1137/0729042.

[21] F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods*, volume 15 of *Springer Series in Computational Mathematics*. Springer-Verlag, New York, 1991. doi:10.1007/978-1-4612-3172-1.

[22] M. C. Cacas, E. Ledoux, G. de Marsily, B. Tillie, A. Barbreau, E. Durand, B. Feuga, and P. Peaudecerf. Modeling fracture flow with a stochastic discrete fracture network: calibration and validation: 1. the flow model. *Water Resources Research*, 26(3):479–489, 1990. doi:10.1029/WR026i003p00479.

[23] X.-C. Cai and M. Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM J. Sci. Comput.*, 21(2):792–797, 1999. doi:10.1137/S106482759732678X.

[24] L. M. Carvalho, L. Giraud, and G. Meurant. Local preconditioners for two-level non-overlapping domain decomposition methods. *Numerical Linear Algebra with Applications*, 8(4):207–227, 2001. doi:10.1002/nla.237.

[25] Y. Chen and G. N. Wells. Multigrid on unstructured meshes with regions of low quality cells, 2024. arXiv:2402.12947, doi:10.48550/arXiv.2402.12947.

[26] C. Chevalier and F. Pellegrini. Pt-scotch: A tool for efficient parallel graph ordering. *Parallel Computing*, 34(6):318–331, 2008. Parallel Matrix Algorithms and Applications. doi:10.1016/j.parco.2007.12.001.

[27] P. Davy, R. Le Goc, and C. Darcel. A model of fracture nucleation, growth and arrest, and consequences for fracture density and scaling. *Journal of Geophysical Research: Solid Earth*, 118(4):1393–1407, 2013. doi:10.1002/jgrb.50120.

[28] P. Davy, R. Le Goc, C. Darcel, O. Bour, J.-R. de Dreuzy, and R. Munier. A likely universal model of fracture scaling and its consequence for crustal hydromechanics. *Journal of Geophysical Research: Solid Earth*, 115(B10), 2010. doi:10.1029/2009JB007043.

[29] D. Demidov. AMGCL: An Efficient, Flexible, and Extensible Algebraic Multigrid Implementation. *Lobachevskii Journal of Mathematics*, 40(5):535–546, 2019. doi:10.1134/S1995080219050056.

[30] D. Demidov. AMGCL – A C++ library for efficient solution of large sparse linear systems. *Software Impacts*, 6:100037, 2020. doi:10.1016/j.simpa.2020.100037.

[31] D. Demidov, L. Mu, and B. Wang. Accelerating linear solvers for Stokes problems with C++ metaprogramming. *Journal of Computational Science*, 49:101285, 2021. doi:10.1016/j.jocs.2020.101285.

[32] C. R. Dohrmann. An approximate BDDC preconditioner. *Numerical Linear Algebra with Applications*, 14(2):149–168, 2007. doi:10.1002/nla.514.

[33] V. Dolean, P. Jolivet, and F. Nataf. *An Introduction to Domain Decomposition Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2015. doi:10.1137/1.9781611974065.

[34] E. Efstathiou and M. J. Gander. Why restricted additive Schwarz converges faster than additive Schwarz. *BIT*, 43:945–959, 2003. doi:10.1023/B:BITN.0000014563.33622.1d.

[35] *GAMG*: Geometric algebraic multigrid (amg) preconditioner. https://petsc.org/release/manualpages/PC/PCGAMG/, https://petsc.org/release/manualpages/PC/PCGAMG/.

[36] G.N. Gatica. *A simple introduction to the mixed finite element method. Theory and applications*. SpringerBriefs in Mathematics. Springer, 2014. doi:10.1007/978-3-319-03695-3.

[37] P.-L. George, H. Borouchaki, F. Alauzet, A. Loseille, P. Laug, and L. Maréchal. *Maillage, modélisation géométrique et simulation numérique 2*. ISTE Group, 2018. URL: https://www.istegroup.com/fr/produit/maillage-modelisation-geometrique-et-simulation-numerique-2/.

[38] L. Giraud, A. Haidar, and L.T. Watson. Parallel scalability study of hybrid preconditioners in three dimensions. *Parallel Computing*, 34(6):363–379, 2008. doi:10.1016/j.parco.2008.01.006.

[39] J. Gopalakrishnan and S. Tan. A convergent multigrid cycle for the hybridized mixed method. *Numerical Linear Algebra with Applications*, 16(9):689–714, 2009. doi:10.1002/nla.636.

[40] H. Al Daas, , P. Jolivet, and T. Rees. Efficient Algebraic Two-Level Schwarz Preconditioner for Sparse Matrices. *SIAM Journal on Scientific Computing*, 45(3):A1199–A1213, 2023. doi:10.1137/22M1469833.

[41] H. Hoteit and A. Firoozabadi. An efficient numerical model for incompressible two-phase flow in fractured media. *Advances in Water Resources*, 31(6):891–905, 2008. doi:10.1016/j.advwatres.2008.02.004.

[42] *hypre*: High performance preconditioners. https://llnl.gov/casc/hypre, https://github.com/hypre-space/hypre.

[43] P. Jolivet, F. Hecht, F. Nataf, and C. Prud'homme. Scalable domain decomposition preconditioners for heterogeneous elliptic problems. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2503210.2503212.

[44] P. Jolivet, J.E. Roman, and S. Zampini. KSPHPDDM and PCHPDDM: Extending PETSc with advanced Krylov methods and robust multilevel overlapping Schwarz preconditioners. *Computers and Mathematics with Applications*, 84:277–295, 2021. doi:10.1016/j.camwa.2021.01.003.

[45] P. Jolivet and P.-H. Tournier. Block iterative methods and recycling for improved scalability of linear solvers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '16. IEEE Press, 2016.

[46] G. Karypis and V. Kumar. Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. Technical report, Univ. of Minnesota, Dept. of Computer Sci. and Engr., 01 1997.

[47] G. Karypis, K. Schloegel, and V. Kumar. ParMETIS: Parallel graph partitioning and sparse matrix ordering library. Technical report, Univ. of Minnesota, Dept. of Computer Sci. and Engr., 01 1997.

[48] M. Kern, G. Pichot, and D. Zegarra Vasquez. Mathematical and numerical analysis of the mixed formulation of single phase flow in three-dimensional fractured porous media. Preprint, 2025. URL: https://inria.hal.science/hal-05029638.

[49] M. Kern, G. Pichot, and D. Zegarra Vasquez. Performance of algebraic preconditioners for large-scale simulations of single-phase flow in three-dimensional fractured porous media. Preprint, 2025. URL: https://inria.hal.science/hal-05029652.

[50] P. Laug and G. Pichot. Mesh Generation and Flow Simulation in Large Tridimensional Fracture Networks. In S. Carillo, C. Conti, D. Mansutti, F. Pitolli, and R. M. Spitaleri, editors, *IMACS Series in Computational and Applied Mathematics*, volume 22, pages 71–80, 2019. URL: https://hal.archives-ouvertes.fr/hal-02102811.

[51] S. Legrand, T. Martinez, and G. Pichot. pruners. *Git Repository*, 2022. URL: https://gitlab.inria.fr/pruners/pruners.

[52] S. B. Lunowa, I. S. Pop, and B. Koren. Linearized domain decomposition methods for two-phase porous media flow models involving dynamic capillarity and hysteresis. *Computer Methods in Applied Mechanics and Engineering*, 372:113364, 2020. doi:10.1016/j.cma.2020.113364.

[53] J. Mandel, B. Sousedík, and C. R. Dohrmann. Multispace and multilevel BDDC. *Computing*, 83(2):55–85, 2008. doi:10.1007/s00607-008-0014-7.

[54] V. Martin, J. Jaffré, and J. E. Roberts. Modeling fractures and barriers as interfaces for flow in porous media. *SIAM Journal on Scientific Computing*, 26(5):1667–1691, 2005. doi:10.1137/S1064827503429363.

[55] The Muelu Project Team. *The Muelu Project Website*.

[56] B. Nœtinger. A quasi steady state method for solving transient Darcy flow in complex 3D fractured networks accounting for matrix to fracture flow. *Journal of Computational Physics*, 283:205–223, 2015. doi:10.1016/j.jcp.2014.11.038.

[57] F. Pellegrini. Scotch and PT-Scotch Graph Partitioning Software: An Overview. In U. Naumann and O. Schenk eds., editors, *Combinatorial Scientific Computing*, page 34. Chapman and Hall/CRC, 2012.

[58] F. Pellegrini and J. Roman. SCOTCH: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs. In H. Liddell, A. Colbrook, B. Hertzberger, and P. Sloot (eds), editors, *High-Performance Computing and Networking. HPCN-Europe 1996. Lecture Notes in Computer Science.*, volume 1067, pages 493—-498. Springer, Berlin, Heidelberg, 1996. doi:10.1007/3-540-61142-8_588.

[59] L. Poirel. *Algebraic Domain Decomposition Methods for Hybrid (direct/iterative) Solvers*. Phd thesis, University of Bordeaux (France), 2018. URL: https://theses.hal.science/tel-03555822v2.

[60] Y. Saad and M. H. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986. doi:10.1137/0907058.

[61] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, and R. Scheichl. Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps. *Numerische Mathematik*, 126(4):741–770, 2014. doi:10.1007/s00211-013-0576-y.

[62] A. Toselli and O. Widlund. *Domain decomposition methods—algorithms and theory*, volume 34 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2005. doi:10.1007/b137868.

[63] M. Vohralík, J. Maryška, and O. Severýn. Mixed and nonconforming finite element methods on a system of polygons. *Applied Numerical Mathematics. An IMACS Journal*, 57(2):176–193, 2007. doi:10.1016/j.apnum.2006.02.005.

[64] T. Wildey, S. Muralikrishnan, and T. Bui-Thanh. Unified geometric multigrid algorithm for hybridized high-order finite element methods. *SIAM Journal on Scientific Computing*, 41(5):S172–S195, 2019. doi:10.1137/18M1193505.