Computational Geosciences (2021) 25:243–265
https://doi.org/10.1007/s10596-020-10002-5

ORIGINAL PAPER

PorePy: an open-source software for simulation of multiphysics processes in fractured porous media

Eirik Keilegavlen[1] · Runar Berge[1] · Alessio Fumagalli[1,2] · Michele Starnoni[1,3] · Ivar Stefansson[1] · Jhabriel Varela[1] · Inga Berre[1]

Received: 26 August 2019 / Accepted: 10 September 2020 / Published online: 14 October 2020

Abstract

Development of models and dedicated numerical methods for dynamics in fractured rocks is an active research field, with research moving towards increasingly advanced process couplings and complex fracture networks. The inclusion of coupled processes in simulation models is challenged by the high aspect ratio of the fractures, the complex geometry of fracture networks, and the crucial impact of processes that completely change characteristics on the fracture-rock interface. This paper provides a general discussion of design principles for introducing fractures in simulators, and defines a framework for integrated modeling, discretization, and computer implementation. The framework is implemented in the open-source simulation software PorePy, which can serve as a flexible prototyping tool for multiphysics problems in fractured rocks. Based on a representation of the fractures and their intersections as lower-dimensional objects, we discuss data structures for mixed-dimensional grids, formulation of multiphysics problems, and discretizations that utilize existing software. We further present a *Python* implementation of these concepts in the PorePy open-source software tool, which is aimed at coupled simulation of flow and transport in three-dimensional fractured reservoirs as well as deformation of fractures and the reservoir in general. We present validation by benchmarks for flow, poroelasticity, and fracture deformation in porous media. The flexibility of the framework is then illustrated by simulations of non-linearly coupled flow and transport and of injection-driven deformation of fractures. All results can be reproduced by openly available simulation scripts.

Keywords Fractured reservoirs · Mixed-dimensional geometry · Numerical simulations · Multiphysics · Discrete fracture matrix models · Open-source software · Reproducible science

1 Introduction

Simulation of flow, transport, and deformation of fractured rocks is of critical importance to several applications such as subsurface energy extraction and storage and waste disposal. While the topic has received considerable attention in the last decade, the development of reliable simulation tools remains a formidable challenge. Many reasons can be given for this; we here pinpoint four possible causes: First, while natural fractures are thin compared to the characteristic length of the domains of interest, their extent can span the entire domain [1]. The high aspect ratio makes the geometric representation of fractures in the simulation model challenging. Second, the strongly heterogeneous properties of fractures compared to the matrix with respect to flow and mechanics call for methods that can handle strong parameter discontinuities as well as different governing physics for the fractures and the matrix, see for instance [2–4]. Third, phenomena of practical interest tend to involve multiphysics couplings, such as interaction between flow, temperature evolution, geo-chemical effects, and fracture deformation [5]. Correspondingly, there is an ongoing effort to develop and introduce multiphysics

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10596-020-10002-5) contains supplementary material, which is available to authorized users.

✉ Eirik Keilegavlen
Eirik.Keilegavlen@uib.no

[1] Department of Mathematics, University of Bergen, Pb 7800, 5020 Bergen, Norway

[2] Present address: MOX Laboratory, Department of Mathematics, Politecnico di Milano, via Bonardi 9, 20133 Milan, Italy

[3] Present address: Department of Environment, Land and Infrastructure Engineering, Politecnico di Torino, Turin, Italy

couplings within simulation models [6]. Fourth, fracture networks have highly complex intersection geometries, which must be accounted for in the simulation models. Although the geometry of the walls of individual fractures can be complex by themselves, we will not consider this in any detail, but rather assume that averaged apertures are available at the scale of discretizations.

Traditionally, simulation of flow-driven dynamics in fractured media has been based on two conceptual models. The first is the upscaled representation, where the fracture network geometry and dynamical processes taking place in the network are replaced by equivalent continuum models, which resemble those used in non-fractured porous media. As these models do not resolve the fracture geometry, they are computationally efficient, and have been extended to cover a wide range of multiphysics couplings, as exemplified by the TOUGH2 family of codes [7] as well as PFLOTRAN [8]. The accuracy of the simulations is however highly dependent on the quality of the upscaled model, which in turn depends on the fractured domain's resemblance of a continuous medium with respect to the nature of the physical processes. In practice, the upscaling process ranges from treatable by analytical means for simple fracture geometries and dynamics [9, 10], to extremely challenging in the case of multiphysics couplings and complex fracture geometries [11, 12].

The second traditional class of models, known as the discrete fracture network (DFN) models, is constructed using an explicit representation of the fracture network in the simulation model, while ignoring the surrounding rock mass. The models combine highly accurate representation of dynamics in the fractures with computational efficiency from not having to deal with the rock matrix. DFN simulation models with a high level of sophistication have been developed, notably for coupled flow and transport, see for instance [13–15]. By themselves, DFN models cannot represent processes outside the fracture network; however, the models can be combined with continuum models to achieve fracture-matrix couplings.

The respective limitations of continuum and DFN models have, over the last decade, led to an increased interest in the class of discrete fracture matrix (DFM) models. In DFM models, the fractures are sorted in two classes according to their importance for the dynamics in question [16]. The most important fractures are represented explicitly, while upscaled models are applied for the remaining fractures and the host rock. As such, DFM models represent a flexible compromise between upscaling and explicit representations. The models can represent governing equations in the rock matrix, fractures, and generally also in the intersections between fractures. For computational efficiency, it is common to represent fractures and their intersections as lower-dimensional objects embedded in the three-dimensional rock matrix [17, 18]. We refer to such representation as a mixed-dimensional model [19], and conversely refer to a model of a domain where only a single dimension is considered fixed dimensional.

DFM models can further be divided into two subgroups, according to whether they explicitly represent the fracture surfaces in the computational grid [16]. Models that apply non-conforming gridding include the embedded discrete fracture matrix model (EDFM) [20], and extended finite element methods (XFEM) [21, 22]. These methods avoid the complexities of conforming grid generation discussed below, but must instead incorporate the fracture-matrix interaction in what becomes complex modifications of the numerical method for XFEM [23], or by constructing an upscaled representation, e.g., [24], where the latter approach faces challenges reminiscent of those in continuum-type models. For this reason, our interest herein is DFM methods with conforming grids. Construction of these grids can be challenging for complex fracture networks, particularly in 3d, and the high cell count that may result can put limits in the amount of fractures that can be explicitly represented. Nevertheless, this type of DFM models has been developed for flow and transport, as well as mechanics and poroelasticity, and the explicit representation is particularly useful when the fractures deform. Simulation models that incorporate DFM principles include DuMuX [25], CSMP [26], MOOSE-FALCON [27, 28], OpenGeoSys [29], and Flow123d [30].

The utility of a rapid prototyping framework is illustrated by the wide usage of the Matlab Reservoir Simulation Toolbox (MRST) [31, 32], mainly for non-fractured porous media. Similarly, research into strongly coupled processes in mixed-dimensional geometries will benefit from software of similar flexibility and with a structure tailored to the specific challenges related to fractured porous media.

The goal of this paper is twofold: First, we review challenges related to design of simulation frameworks for multiphysics couplings in mixed-dimensional geometries. Our aim is to discuss design choices that must be made in the implementation of any DFM simulator, including data structures for mixed-dimensional geometries, and representation and discretization of multiphysics problems. Second, we describe a framework for integrated modeling, discretization, and implementation, and an open-source software termed PorePy adhering to this framework. Key to our approach is a decomposition of the geometry into separate objects for rock matrix, individual fractures, and fracture intersections. Governing equations can then be defined separately on each geometric object, as well as on the connection between the objects. This allows for significant code reuse from the discretization of fixed-dimensional problems; thus, our design principles are also applicable to more general PDE software frameworks, such as FEniCS [33], Dune [34], and FireDrake [35]. Furthermore, for scalar and vector elliptic problems

(flow and deformation), the models rest on a solid mathematical formulation [36–38].

Built on the mixed-dimensional geometry, PorePy offers several discretization schemes for mathematical models of common processes, such as flow, transport, and mechanical deformation. Multiphysics couplings are easily formulated, and their discretization depends on the availability of appropriate discretization schemes. Moreover, the framework allows for different geometric objects to have different primary variables and governing equations. The software can be used for linear and non-linear problems, with the latter treated by automatic differentiation. PorePy offers automatic gridding of fractured domains in 2d and 3d, relying on the third-party software Gmsh [39] to construct the grid. PorePy is fully open-source (see www.github.com/pmgbergen/porepy) and is released under the GNU General Public License (GPL) version 3.

The paper is structured as follows: In Section 2, we present the principles whereupon we have built the mixed-dimensional framework in PorePy. Section 3 presents models for physical processes central to fractured porous media: single-phase flow, heat transport, and poroelastic rock deformation coupled with fracture deformation modeled by contact mechanics. The implementation of PorePy is presented in Section 4. In Section 5, we benchmark our approach and the PorePy library against well-established test cases. In Section 6, we present two complex applications to illustrate the potential of the framework with respect to advanced physical processes, followed by conclusions in Section 7.

2 Design principles for mixed-dimensional simulation tools

Developing a simulation model for a specific process in mixed-dimensional media requires three main ingredients: A representation of the mixed-dimensional geometry, governing equations for dynamics within and between the geometric objects (rock matrix, fractures, and fracture intersections), and a strategy for discretization and assembly of the equations on the geometry. This in turn leads to decisions on how much of the mixed-dimensional geometry to represent, which type of couplings between different geometric objects to permit, and how to establish communication between the geometric objects.

In this section, we discuss principles for modeling of coupled processes between dimensions in a general context of fractured rocks, together with representation of the geometry in a continuous and discrete setting. The general discussion herein is supplemented by concrete examples of modeling of the important processes presented in Section 3, while discretizations and implementation are discussed in Section 4.

2.1 Representation of a mixed-dimensional geometry

We consider the representation of a fracture network embedded in a 3d domain. The dimension of the fractures is reduced to 2. Similarly, fracture intersections are reduced to 1d objects and intersections of intersection lines to 0d, producing a hierarchy of objects of dimensions 0 to 3. For a fracture network in a 2d domain, the natural simplification applies, i.e., fractures will be objects of dimension 1 and intersections objects of dimension 0. An important modeling choice is which parts of the geometry to represent in the model. We emphasize that, as our focus herein is DFM models with explicit fracture representation, it is assumed that at least the dominating fractures and the matrix will be explicitly represented in the simulation model, and furthermore that the simulation grid will conform to the fractures.

We distinguish between two approaches for the representation of the fracture geometry: The first explicitly represents the full hierarchy of geometric objects (3d–0d). However, for many processes, one can to a good approximation assume that the main dynamics take place in the matrix or in the fractures, while objects of co-dimension more than 1 (intersection lines and points) mainly act as transition zones between fractures. This observation motivates the second approach: The matrix and fractures are represented explicitly, together with some model for direct fracture-fracture interaction.

Representation only of matrix and fractures and not the intersections in some sense constitutes the minimal modification to an existing fixed-dimensional model and has been a popular choice, e.g., for flow and transport problems [40]. The strategy has also been taken a long way towards practical applications, see for instance [41]. There are however drawbacks, notably in the treatment of fracture intersections: Without explicit access to the intersection objects, modeling of interaction between two fractures can be challenging. As an example, for flow, the model does not allow for specifying the permeability of the intersection between two fractures. Significantly, the difficulties tend to increase with increasing complexity of the dynamics, such as countercurrent flow due to gravity and capillary forces, and when transitioning from 2d domains to 3d, i.e., the dimension of the intersections increases from zero to one. This has important consequences for model and method development, as issues related to ad hoc treatment of intersection dynamics may not manifest until relatively late in the development process. For these reasons, we prefer the first approach, where all geometric objects are treated (or "represented") equally, independent of their dimension.

To illustrate our geometry representation, consider Fig. 1a showing three fractures that intersect pairwise along three

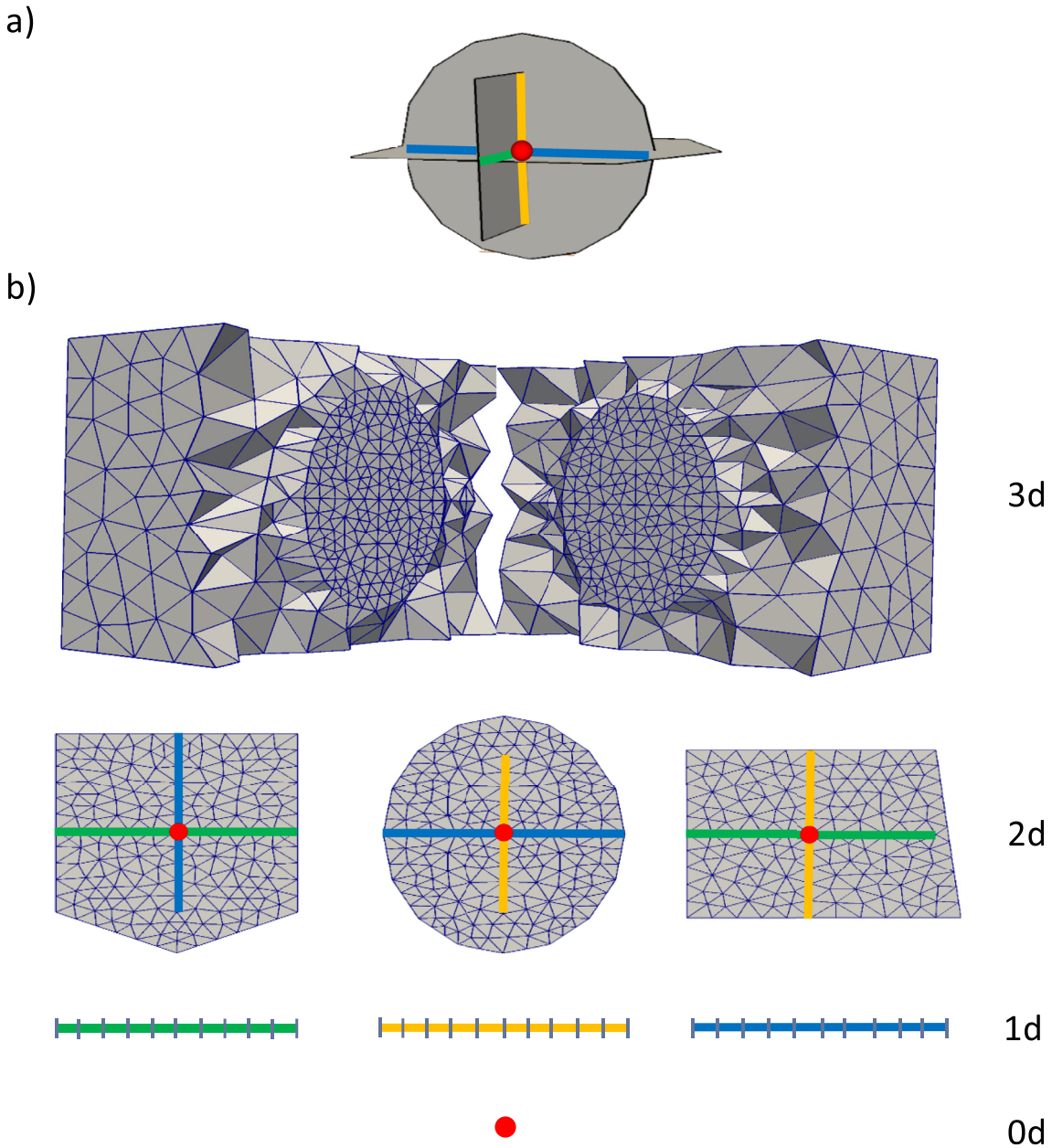

Fig. 1 Conceptual illustration of a fracture network, including grids and lower-dimensional representation. (**a**) Fracture network, the rock matrix is not visualized. (**b**) Grids of all subdomains. Fracture intersections (1d) are represented by colored lines, the 0d grid by a red circle. The 3d grid is cut to expose the circular fracture

lines, which in turn intersect in a point. The fracture network thus defines a set of objects of dimensions $\{0, 1, 2\}$, while the surrounding host medium (not shown) is 3d. We shall refer to each object as a subdomain and denote a generic subdomain by Ω_i. Note that all subdomains of dimension less than 3 are embedded in at least one subdomain of one dimension more, for instance, all lines in the geometry lie on at least two fracture surfaces.

Figure 1b shows the computational grid constructed for each subdomain. The grid on each subdomain conforms to any lower-dimensional subdomains embedded within it, illustrated by the faces in the 3d grid that match the circular fracture. We will discuss grid construction in more detail in Section 4.1.

To finalize the description of the geometry, we introduce the notation for an interface between two subdomains. With reference to Fig. 2, we denote by Ω_h and Ω_l two subdomains one dimension apart so that Ω_l is embedded in Ω_h, and let $\partial_j\Omega_h$ be the part of the boundary of Ω_h that geometrically coincides with Ω_l. Furthermore, we introduce the interface Γ_j on the boundary between $\partial\Omega_h$ and Ω_l. From the dimension reduction, it follows that Γ_j, Ω_l, and $\partial_j\Omega_h$ all coincide geometrically. For completeness, we note that the mathematical framework [36] on which our models are based considers the two sides of Ω_l as

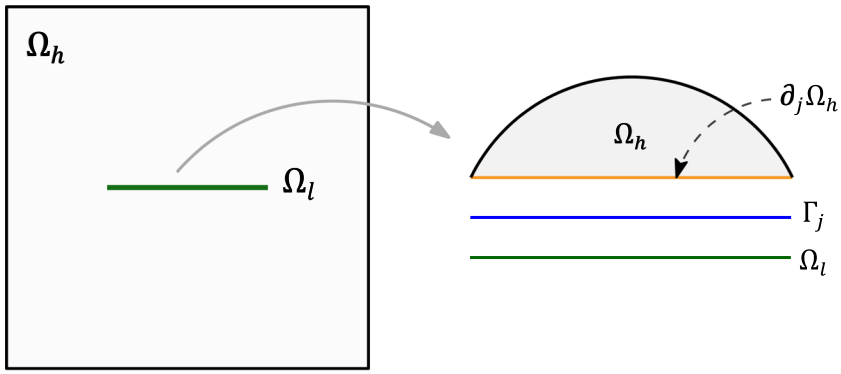

Fig. 2 Mixed-dimensional geometric objects. A higher-dimensional subdomain Ω_h is connected to a lower-dimensional subdomain Ω_l through the interface Γ_j. The part of the boundary of Ω_h geometrically coinciding with Ω_l is denoted by $\partial_j\Omega_h$. The interface Γ_k on the lower side of Ω_l is not shown

different interfaces, Γ_j and Γ_k. Throughout, we will let Γ_j denote a generic interface and use the triplet (Γ_j, Ω_h, Ω_l) to represent an interface and its higher- and lower-dimensional neighbor.

2.2 Permissible coupling structures for mixed-dimensional processes

For modeling purposes, it is important to establish which types of couplings between variables on subdomains and interfaces are permitted. In our framework, we impose the following constraints on the modeling of dynamic processes:

1. There is only coupling between subdomains that are exactly one dimension apart.
2. Interaction between subdomains is formulated as a model on the interface between the subdomains.
3. A model on an interface can depend on variables on the interface and the immediate subdomain neighbors, but not on variables associated with other subdomains or interfaces.

These choices have two important consequences: First, our framework explicitly rules out direct 3d-1d couplings. Second, our model does not permit direct coupling between objects of the same dimension, say, two fractures; the communication must go via a lower- or higher-dimensional object. On the other hand, the imposed constraints make the structure of the equations on a subdomain relatively simple, as the dynamics depend only on variables internal to the subdomain and on neighboring interfaces.

In some cases, it can be of interest to also consider couplings between subdomains of equal dimension, for instance to implement domain decomposition solvers. This can be realized by a secondary partitioning of the subdomains. When such a strategy is applied, the above constraints should be applied only on the interface between subdomains of different dimensions. On interfaces between subdomains of the same dimension, standard continuity conditions can be applied.

3 Model problems

In this section, we use the modeling framework defined in Section 2 to present three sets of governing equations, each of which is of high relevance for fractured porous media: the elliptic pressure equation, fully coupled flow and transport, and fracture deformation coupled with poroelastic deformation of the host medium. Since most of the involved fixed-dimensional processes are well established, our main purpose is to apply the modeling framework described in Section 2 to the mixed-dimensional setting.

We introduce the following notation for variables and subdomains: Variables in a generic subdomain Ω_i are marked by the subscript i, while the subscript j identifies interface variables on Γ_j. For a subdomain Ω_i, the set of neighboring interfaces is split into interfaces towards subdomains of higher dimensions, denoted $\widehat{S}_i$, and interfaces towards subdomains of lower dimensions, denoted by $\check{S}_i$ (see Fig. 3).

Communication between an interface and its neighboring subdomains is handled by projection operators. In the subsequent parts, we will apply four different classes of projections. We indicate the mapping from an interface to the related subdomains by Ξ, with a subscript indicating the index of the interface and a superscript denoting the index of the subdomain, as illustrated in Fig. 4. We also introduce the projection operators from subdomains neighboring of an interface to the interface itself, denoted by the symbol Π with the same convention as before for sub- and superscripts. The actual definition of these objects is scope-dependent and will be specified when needed. The construction of the projection needs to consider the nature of the variable to project, being of intensive or extensive kind, that is, whether the projections should average or sum the variables, respectively.

3.1 Flow in fractured media

We first consider incompressible flow in mixed-dimensional geometries, where we assume a Darcy-type relation between

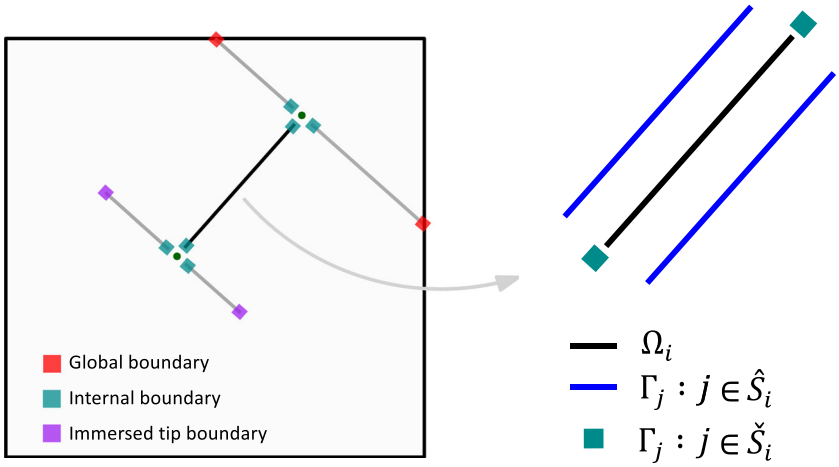

Fig. 3 Intersecting fractures, interfaces, and types of boundary conditions. The 2d domain contains three fractures (1d lines) that intersect in two intersection points (dots). The fractures have three types of boundaries: internal (green squares), immersed tips (purple squares), and endings at the external boundary (red squares). A close-up of the black fracture Ω_i shows the interfaces associated with its higher-dimensional (blue lines) and lower-dimensional (green squares) neighboring subdomains. The sets of such interfaces are denoted respectively by $\hat{S}_i$ and $\check{S}_i$

the flux and the pressure gradient in all subdomains. The model has been presented several times before, see, e.g., [2, 42, 43].

First, consider a domain with a single interface Γ_j with neighboring subdomains Ω_h and Ω_l. In addition to the pressure p_i and flux q_i in each subdomain, we denote the flux on Γ_j by λ_j and formally write $\lambda_j = \Pi_j^h \, tr \, q_h \cdot n_h$, with n_h the unit normal on $\partial_j\Omega_h$ pointing from Ω_h to Ω_l, and tr a suitable trace operator mapping from Ω_h to $\partial_j\Omega_h$, referring to Fig. 4. The strong form of the Darcy problem for Ω_l reads: find (q_l, p_l) such that

$$\begin{aligned} q_l + \frac{\mathcal{K}_l}{\mu_l}\nabla p_l &= 0, \\ \nabla \cdot q_l - \Xi_j^l \lambda_j &= f_l \end{aligned}, \tag{3.1}$$

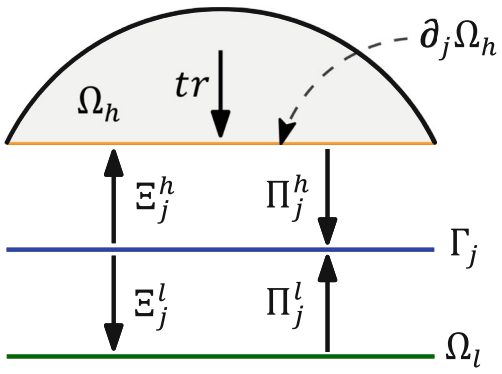

Fig. 4 Representation of a generic coupling between two subdomains. An interface Γ_j is coupled to a higher-dimensional subdomain Ω_h and a lower-dimensional subdomain Ω_l. The projection operators are denoted by Ξ (interface to subdomains) and Π (subdomains to interface) with subscripts indicating the interface and superscripts indicating the subdomain. The trace operator tr maps quantities from Ω_h to its boundary $\partial_j\Omega_h$

where the differential operators are defined on the tangent space of Ω_l and Ξ_j^l maps from Γ_j to Ω_l. We have indicated with f_l a source or sink term, μ_l is the fluid viscosity, while $\mathcal{K}_l$ represents the effective tangential permeability tensor scaled by the aperture as described in [42]. An analogous problem is written for (q_h, p_h), with the exception that $\Xi_j^h \lambda_j$ is mapped to a boundary condition on $\partial_j\Omega_h$,

$$q_h \cdot n_h|_{\partial_j\Omega_h} = \Xi_j^h \lambda_j. \tag{3.2}$$

The flux λ_j is given by an interface condition on Γ_j, which reads

$$\lambda_j + \frac{\kappa_j}{\mu_j}\left(\Pi_j^l p_l - \Pi_j^h tr\, p_h\right) = 0. \tag{3.3}$$

Here, κ_j indicates the normal effective permeability. Equation (3.3) can be seen as a Darcy law in the normal direction of Γ_j. Different types of boundary conditions can be imposed on the external boundary of Ω_h and Ω_l. Moreover, we impose null flux if Ω_l has an immersed tip boundary.

The extension to problems with many subdomains is now immediate: The flux on an interface is still formulated in terms of variables on its two neighboring subdomains, while for a subdomain Ω_i summation over all neighboring interfaces gives the problem: Find (q_i, p_i) so that

$$\begin{aligned} q_i + \frac{\mathcal{K}_i}{\mu_i}\nabla p_i &= 0, \\ \nabla \cdot q_i - \textstyle\sum_{j\in S} \Xi_j^i \lambda_j &= f_i, \\ q_i \cdot n_i|_{\partial_j\Omega_i} &= \Xi_j^i \lambda_j \quad \forall j \in \check{S}_i \end{aligned} \tag{3.4}$$

In the case of $d = 0$, most of the above terms are void, and we are left with the balance between the source term and fluxes from higher dimensions, while for the case $d = 3$, the term involving interface fluxes from higher dimensions is void.

3.2 Fully coupled flow and transport

We next turn to modeling of fully coupled flow and transport, as an example of a multiphysics problem with variable coupling within and between subdomains. We consider a single-phase flow of an incompressible fluid with two components that mix ideally. We denote by c_i the mass fraction of a component associated with Ω_i; the closure relation for the mass fractions implies that we can calculate the other value by $1 - c_i$. The governing equation of the fluid is given by Darcy's law and the fluid mass conservation as in Eq. (3.4). However, we let the viscosity of the fluid depend on the mass fraction,

$$\mu_i = \mu_i(c_i). \tag{3.5}$$

The conservation equations for the components can be formulated as

$$\phi_i \frac{\partial c_i}{\partial t} + \nabla \cdot (c_i q_i - \mathcal{D}_i \nabla c_i) - \sum_{j \in S_i} \Xi_j^i \left(\eta_j + \beta_j \right) = g_i. \tag{3.6}$$

Here, ϕ_i represents the effective porosity, $\mathcal{D}_i$ is the effective diffusivity, and g_i denotes sources and sinks. A sum of advective, η_j, and diffusive, β_j, fluxes from the higher-dimensional domains is included in the conservation equation. As for the flow problem, flow over lower-dimensional interfaces Γ_j, $j \in \check{S}_i$, enters as Neumann boundary conditions. We note that the governing equations are coupled via the mass fraction dependency of viscosity and the presence of the Darcy flux in the advective transport.

Let us now consider the interaction between two neighboring subdomains Ω_h and Ω_l via the common interface Γ_j. The flow over Γ_j, denoted by λ_j, is given by Eq. (3.3), where the interface viscosity μ_j is modeled as a function of the mean of the mass fractions on the two sides,

$$\mu_j = \mu_j \left(\frac{\Pi_j^l c_l + \Pi_j^h tr\, c_h}{2} \right). \tag{3.7}$$

The component flux over Γ_j is again governed by an advection-diffusion relation: The diffusion term β_j is, in analogy with the corresponding term for the Darcy flux, given by

$$\beta_j + \delta_j \left(\Pi_j^l c_l - \Pi_j^h tr\, c_h \right) = 0, \tag{3.8}$$

with δ_j representing the effective diffusivity over the interface Γ_j. For the advective term η_j, we introduce an upstream-like operator based on the Darcy interface flux:

$$Up(c_h, c_l; \lambda_j) = \begin{cases} \Pi_j^h tr\, c_h, & \text{if } \lambda_j \geq 0 \\ \Pi_j^l c_l, & \text{if } \lambda_j < 0. \end{cases} \tag{3.9}$$

With this, the advective interface flux η_j is given by the relation

$$\eta_j - \lambda_j Up(c_h, c_l; \lambda_j) = 0. \tag{3.10}$$

Finally, global boundary conditions are imposed in the standard way for elliptic and advection-diffusion problems, see, e.g., [44]. Equations (3.5)–(3.10) define the governing equations in all subdomains and on all interfaces, with the exception of 0d domains, where the diffusion operator again is void.

3.3 Poroelastic fracture deformation by contact mechanics

Our final set of model equations considers poroelastic deformation of a fractured medium, where the fractures may open or, if the frictional forces are insufficient to withstand tangential forces on the fracture surface, undergo slip. This process is important in applications such as geothermal energy extraction and CO_2 storage. Modeling of the process is non-trivial due to (i) the coupled poroelastic processes, (ii) the heterogeneous governing equations between subdomains, (iii) the need to use non-standard constitutive laws to relate primary variables during sliding, and (iv) the non-smooth behavior of the constitutive laws in the transition between sticking and sliding and between open and closed fractures. Modeling of this process is an active research field, see, e.g., [45–47], and thus represents an example where the availability of a flexible prototyping framework is highly useful. Due to the complexity in deformation of intersecting fractures, we limit our exposition to media with non-intersecting fractures.

Flow and deformation in the rock matrix, represented by the subdomain Ω_h, are governed by Biot's equations for poroelasticity [48].

$$\begin{aligned} &\nabla \cdot (\mathcal{C}_h \nabla_s u_h - \alpha_h p_h I) = b_h, \\ &\alpha_h \frac{\partial (\nabla \cdot u_h)}{\partial t} + \theta_h \frac{\partial p_h}{\partial t} - \nabla \cdot \left(\frac{\mathcal{K}_h}{\mu_h} \nabla p_h \right) = f_h \end{aligned} \tag{3.11}$$

Here, the first equation represents conservation of momentum, with the acceleration term neglected, while the second equation expresses conservation of mass. The primary variables are the displacement, u_h, and the fluid pressure, p_h. The stiffness matrix $\mathcal{C}_h$ can for linear isotropic media be expressed purely in terms of the first and second Lamé parameters, and the elastic stress can be computed as

$$\sigma_h = \mathcal{C}_h \nabla_s u_h,$$

where ∇_s is the symmetric gradient. Furthermore, α_h is the Biot constant, I the second-order identity tensor, b_h denotes body forces, and θ_h the effective storage term. We also assume boundary conditions are given on the global boundary.

Next, to model relative motion of the fracture walls, it is necessary to consider both interfaces between Ω_h and Ω_l. In a slight abuse of notation, we will let u_j denote the displacement variable on both interfaces. We emphasize that u_j is a vector in $\mathbb{R}^n$, that is, it represents the displacement in both the tangential and normal direction of Ω_l. We will require continuity between u_h and u_j, expressed as $\Pi_j^h tr\, u_h = u_j$, where we recall that the trace operator maps to $\partial_j\Omega_h$. We also introduce the jump in displacement, $[\![u_j]\!]$, between the two interfaces on opposing sides of Ω_l (see Fig. 5). The jump is decomposed into the tangential jump $[\![u_j]\!]_\tau$ and the normal jump $[\![u_j]\!]_n$.

The mechanical state in Ω_l is described by the contact traction σ_l, which also is a vector in $\mathbb{R}^n$, with normal and tangential components $\sigma_{l,n}$ and $\sigma_{l,\tau}$, respectively. Our model also includes fluid flow in the fracture Ω_l, which is governed by conservation of mass

$$\frac{\partial}{\partial t}\left(a([\![u_j]\!])\right) + \theta_l \frac{\partial p_l}{\partial t} - \nabla\cdot\left(\frac{\mathcal{K}_l}{\mu_l}\nabla p_l\right) - \Xi_j^l \lambda_j = f_l. \quad (3.12)$$

Here, the time derivative of the aperture $a([\![u_j]\!]) = a_0 - [\![u_j]\!]_n$ represents changes in the available volume due to changes in the displacement jump, with a_0 denoting the residual hydraulic aperture. The negative sign on the normal jump is related to the sign convention in (3.14) below. As in the previous sections, the relation between the fluid pressures in Ω_h and Ω_l is governed by a flux law of the type (3.3).

The relation between σ_l and $[\![u_j]\!]$ is modeled by borrowing techniques from contact mechanics as summarized here (for a full discussion, see [49]). Balance of tractions between the poroelastic stress in Ω_h and the contact traction in Ω_l is for the two sides expressed as

$$\begin{aligned}\Pi_{j_1}^h n_h\cdot(\sigma_h - \alpha_h p_h I) &= \Pi_{j_1}^l \sigma_l - \left(\Pi_{j_1}^h n_h\right)\cdot\left(I\Pi_{j_1}^l \alpha_l p_l\right)\\ \Pi_{j_2}^h n_h\cdot(\sigma_h - \alpha_h p_h I) &= -\Pi_{j_2}^l \sigma_l - \left(\Pi_{j_2}^h n_h\right)\cdot\left(I\Pi_{j_2}^l \alpha_l p_l\right)\end{aligned} \quad (3.13)$$

The contact traction is zero whenever the normal displacement jump is nonzero, that is

$$[\![u_j]\!]_n \le 0, \quad \sigma_{l,n} \le 0, \quad [\![u_j]\!]_n \sigma_{l,n} = 0. \quad (3.14)$$

For closed fractures, the motion in the tangential direction is controlled by the ratio between the tangential traction $\sigma_{l,\tau}$ and the maximum available frictional traction $F\sigma_{l,n}$, where F is the friction coefficient. The time derivative of the displacement jump is zero until the frictional traction is overcome; for larger tangential tractions, the time derivative of the displacement jump and tangential traction are parallel:

$$\begin{aligned}&\|\sigma_{l,\tau}\| \le -F\sigma_{l,n},\\ &\|\sigma_{l,\tau}\| < -F\sigma_{l,n} \rightarrow [\![u_j]\!]_\tau = 0,\\ &\|\sigma_{l,\tau}\| = -F\sigma_{l,n} \rightarrow \exists\gamma\in\mathbb{R}, \sigma_{l,\tau} = -\gamma^2 [\![u^j]\!]_\tau.\end{aligned} \quad (3.15)$$

Here $\|\cdot\|$ represents the Euclidean norm, and $[\![u_j]\!]_\tau$ the sliding velocity. We emphasize that the tangential contact conditions are formulated in terms of the contact traction σ_l, with no contribution from the fluid pressure p_l.

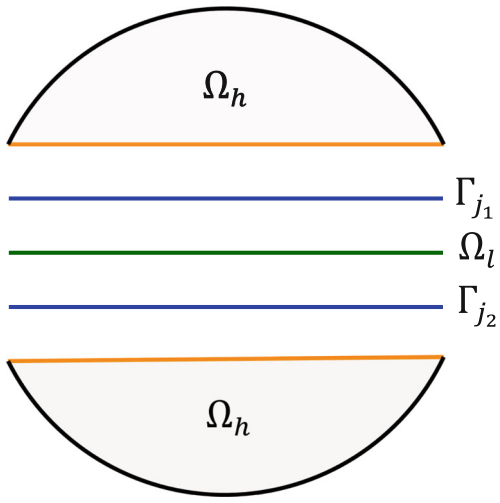

Fig. 5 Illustration of a lower-dimensional domain, Ω_l, that has two interfaces, Γ_{j_1} and Γ_{j_2}, with a higher-dimensional domain, Ω_h

4 Implementation

This section describes the implementation of the mixed-dimensional simulation framework outlined above in the open-source simulator PorePy. Our emphasis is on three topics that are particular to this type of DFM simulation models: Gridding, discretization of subdomain couplings, and how to deal with parameters, variables, and linear systems for multiphysics problems that are defined on an arbitrary number of subdomains and dimensions. The ability to treat these components with relatively simple input is the main distinguishing feature of PorePy, and thus, the section gives an overview of the important properties of the implemented simulator.

Figure 6 displays the main components of PorePy, with emphasis on the mixed-dimensional aspects of the code. The implementation follows the principles of locality of variables and equations described in the previous sections. Specifically, equations and discretizations are assigned on individual subdomains, and the implementation of specific discretization schemes closely resembles that applied to fixed-dimensional problems. Similarly, the stencil of interface couplings is

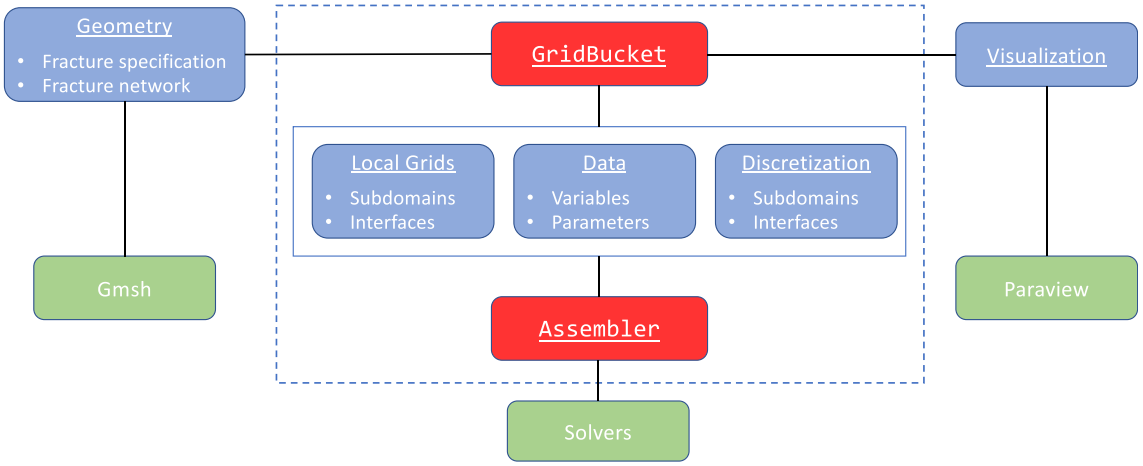

Fig. 6 Outline of the architecture of PorePy: The main mixed-dimensional components are the GridBucket class, which is a combined grid and data manager, and the Assembler class, which acts as a degree of freedom manager. Variables, parameters, and discretizations are local to subdomains and interfaces. Geometry specification and grid construction is handled in part by communication with Gmsh, while visualization is available through export to Paraview. Green boxes represent external dependencies

limited to the interface and the immediate neighboring subdomains. The connection between the subdomains is handled in a top-down manner and implemented in two core classes: The GridBucket class keeps track of the relation between neighboring subdomains and interfaces, and it also acts as a facility for storage of parameters and variables. The Assembler class can be considered a global degree of freedom manager which also has methods for global discretization and assembly. These core mixed-dimensional components are supplemented by functionality for grid construction, assisted by Gmsh, while visualization and linear solvers must be handled by external packages.

A typical workflow for a mixed-dimensional simulation will consist of the following steps:

1. Specify the problem geometry. Use this to create a GridBucket object, that is, a mixed-dimensional grid.
2. On the individual subdomains and interfaces in the GridBucket, specify variables, parameters, and discretizations (thus implicitly define governing equations).
3. Create an Assembler object, use this for initial discretization and assembly of linear system.
4. Solve the mixed-dimensional problem.

Depending on the problem characteristics, the last point can entail non-linear iterations, time stepping, etc.

The rest of this section presents design choices and concrete implementation details of the individual steps. As an illustration of the usage of the resulting simulation framework, Fig. 7 provides an example PorePy code for the setup, discretization, and solution of the mixed-dimensional compressible flow problem. We emphasize that to change the problem geometry, e.g., the fracture network, it is sufficient to change the pink section, while governing equations, parameters, and/or discretization schemes are altered by modifications to the green section. Several examples of the latter are given in Section 5.

4.1 Mixed-dimensional geometry and gridding

Grid construction is one of the main technical bottlenecks for the application of conforming DFM models. The translation of a geometric description of the fracture network into a computational grid consists of three steps: Identification of intersection lines and points, construction of the mixed-dimensional grid, and post-processing of the grid into a format that is suited for the discretization approaches described in Section 4.2. The first and third of these tasks are technically challenging, and one of the strengths of PorePy is that it provides a robust implementation with a simple interface. The second item, grid construction, is a highly advanced research topic in its own; in PorePy, this is handled by a Gmsh backend.

4.1.1 Geometry processing

In PorePy, fractures are described as lines (for 2d domains) or convex planar polygons (in 3d). Curved objects are not supported, as this would significantly complicate the task of identifying intersections; however, piecewise linear approximations are possible. The fractures are specified by their endpoints (in 2d) or vertexes (in 3d). Individual fractures are collected into FractureNetwork2d and FractureNetwork3d classes.

Before passing the fracture network to a gridding software, all fracture intersections must be found. In principle, the computation of fracture intersections is straightforward, following

Section 4.1 / Section 4.2 / Section 4.3 / Section 4.4

```
import porepy as pp
import numpy as np
from scipy.sparse.linalg import spsolve

## Define fractures and fracture network
f1 = pp.Fracture(np.array([[0, 1, 1, 0], [0, 0, 1, 1], [0, 0, 0, 0]]))
f2 = pp.Fracture(np.array([[0, 0, 0, 0], [0, 1, 1, 0], [0, 0, 1, 1]]))
network = pp.FractureNetwork3d([f1, f2])

# Construct GridBucket using prescribed mesh size parameters
gb = network.mesh({'mesh_size_frac': 0.1, 'mesh_size_bound': 1, 'mesh_size_min':0.01})

## Define parameters and discretizations
diffusion_discr = pp.Mpfa('flow')  # Discretization for diffusion term
accumulation_discr = pp.MassMatrix('flow')
for g, d in gb:  # Loop over all subdomain grids (g) and their associated data (d)
    pp.initialize_default_data(g, d, 'flow')  # Default parameters for flow problem
    d[pp.PRIMARY_VARIABLES] = {'p': {'cells': 1}}  # Primary var. p, one dof per cell

    # Assign discretizations for accumulation and diffusion terms
    d[pp.DISCRETIZATION] = {'p': {'accumulation': accumulation_discr,
                                  'diffusion': diffusion_discr}}

for e, d in gb.edges():  # Loop over all interfaces (e) and their associated data (d)
    g_l, g_h = gb.nodes_of_edge(e)   # Get grids of neighboring subdomains
    mg = d['mortar_grid']            # Get hold of mortar grid
    data = {'normal_diffusivity': 1} # Declare interface parameters
    pp.initialize_data(mg, d, 'flow', data)  # Default parameters for flow problem
    d[pp.PRIMARY_VARIABLES] = {'mortar_flux': {'cells': 1}}  # Primary variable
    # The interface discretization has access to associated subdomain discretizations
    interface_discr = pp.RobinCoupling('flow', diffusion_discr, diffusion_discr)
    # Define coupling term through variables and terms/discretizations on the
    # interface and the neighboring subdomains
    d[pp.COUPLING_DISCRETIZATION] = { 'interface_flux':  # identifier of this term
        {g_h: ('p', 'diffusion'),               # variable and term on Omega_h
         g_l: ('p', 'diffusion'),               # variable and term on Omega_l
         e: ('mortar_flux', interface_discr)}}  # variable and term on Gamma_j

## Create object for global assembly and discretization
assembler = pp.Assembler(gb)
assembler.discretize()                 # Call discretize on all local discretizations
A, b = assembler.assemble_matrix_rhs()  # Assemble global discretization matrices

## Solve linear system and export results
p = spsolve(A, b)                 # Solve sparse linear system
assembler.distribute_variable(p)  # Distribute variables to subdomains and interfaces

# Set up an exporter to write the mixed-dimensional pressure field to vtk
paraview_exporter = pp.Exporter(gb, file_name='foo')
# Write data to vtk, ready for import in Paraview.
paraview_exporter.write_vtk('p')
```

Fig. 7 Setup of a full PorePy simulation, illustrated by a mixed-dimensional compressible flow problem solved with a single time step. The background colors indicate different simulation stages, which are discussed in detail in the indicated subsections

for instance [50]. However, to reduce the complexity of the grid construction and limit the number of cells in the resulting grid, it can be useful to alter the geometry to avoid small details, such as almost intersecting fractures. PorePy automatically merges objects that are closer than a user-specified tolerance, and also cuts dangling fracture ends. While such modifications can alter the connectivity of the network, we have found that it is a critical ingredient for dealing with fracture networks that originate from sources that have not removed such small details, for instance networks exported from geological processing software or stochastic fracture network generators.

4.1.2 Gridding

The computational grid should conform to all fractures, and by extension also to their intersection lines and points. This is a difficult problem; however, algorithms [51–53] and high-quality implementations [54, 55] are available. PorePy relies on Gmsh [39] for the grid construction, as this allows for a unified approach in both 2d and 3d domains. While Gmsh allows for a nuanced specification of grid sizes, only a limited set of this functionality is exposed in the PorePy interface: A grid size can be set for the fracture network and the far field; more advanced settings can be accessed by direct manipulations in Gmsh. Still, the specified geometry implicitly sets conditions on the grid size; if the fracture network contains fractures that are close relative to the specified grid size, Gmsh will attempt to construct a grid with reasonable quality, and thereby override the user preferences if necessary.

4.1.3 Construction of grids, mortar grids, and projection operators

The grids provided by Gmsh must be post-processed to be of use for our mixed-dimensional simulations. First, grids for individual subdomains must be extracted. Second, mortar grids must be constructed on the interface between subdomain grids, together with projection operators between the grids. Third, the resulting sets of grids must be arranged in the mixed-dimensional GridBucket.

Subdomains of different dimensions can be identified from Gmsh tags that for each cell identify the geometric object to which the cell belongs (matrix, fracture, or intersection). However, to avoid direct connection between cells that lie on different sides of lower-dimensional objects, faces must be split, and nodes duplicated before the grids are arranged in the GridBucket. This process is illustrated in Fig. 8, which also shows the resulting lower-dimensional grids. Note that while all (d–1)-dimensional faces are split in two, the number of duplicates of a node depends on whether it is located on an intersection, a fracture tip or a global boundary, or in the interior of the subdomain. After this modification, the cells that belong to the same geometric objects are collected into subdomain grids. These are implemented as standard fixed-dimensional grids, so that when a discretization scheme is applied to a subdomain, this is indistinguishable from the traditional fixed-dimensional operation. In this spirit, the grid structure used for individual grids is agnostic to spatial dimension, with an implementation heavily inspired by that of MRST [32].

The mortar grids constructed under post-processing of the Gmsh output are associated with the interfaces. They match with the lower-dimensional grid, and thereby also with the split faces of the higher-dimensional grid. The mortar grids also have methods for the construction of projection matrices between themselves and the lower- and higher-dimensional neighboring subdomains, with separate methods for the mapping of extensive and intensive quantities. Only the lowest order projection operators are available in PorePy, which for matching grids simply identify the split faces of Ω_h with cells in Γ_j, and cells in Γ_j with cells in Ω_i. However, non-matching grids can be introduced by replacing individual subdomain and mortar grids. Specifically, computational speedups can often be achieved by combining fine grids in fractures, which are often the main venue for dynamical processes, with relatively coarse grids in the matrix. During the replacement, the projection operators are automatically updated to account for the resulting non-matching grids.

The individual subdomains and mortar grids are collected in the GridBucket class. This is implemented as a graph, where each subdomain grid Ω_i defines a node, while the interface Γ_j is represented as an edge in the graph, and is identified by the pairing of its neighboring subdomains (Ω_h, Ω_l). In addition to keeping track of geometric information, the GridBucket also provides flexible data storage in the form of dictionaries on subdomains and interfaces. These are used for parameters, discretizations, simulation results, and other data if relevant.

4.2 Primary variables, parameters, and discretization

To define a problem to be discretized in PorePy, one must define primary variables, governing equations, and problem

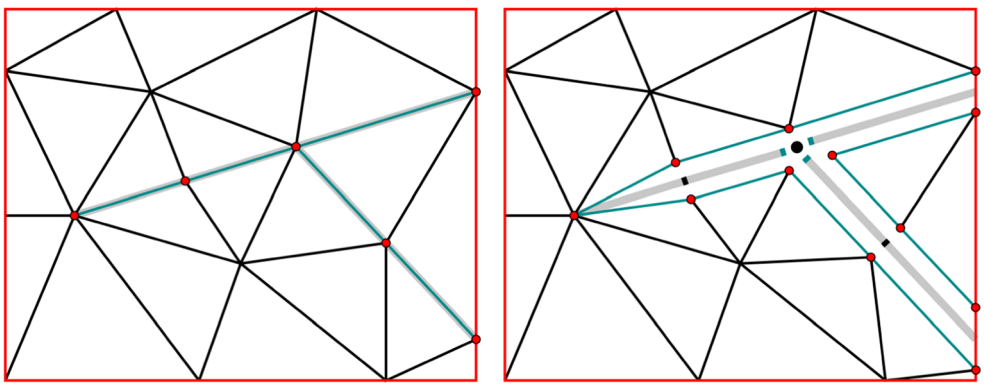

Fig. 8 The process of splitting the faces and nodes of the grid. The faces and nodes of the 2d grid that coincide with the 1d grids (gray lines) are split and define an internal boundary of the grid. Similarly, the faces and nodes of the 1d grids that coincide with the 0d grid (black dot) are split. Note that the split nodes and faces coincide geometrically but have been shifted in the right figure for illustrative purposes

parameters. PorePy is designed to allow for maximum flexibility in these specifications. Variables and parameters are defined on individual subdomains and interfaces. Governing equations are specified in terms of their discretizations: Each variable can be assigned one or several discretizations corresponding to different terms in the equation. As with the variable specification, discretizations are specified locally on subdomains and interfaces, thus heterogeneous governing equations or discretization schemes can readily be assigned. It is up to the user to ensure that the specified combination of variables, equations, and discretizations is mathematically well posed on the given mixed-dimensional grid.

In terms of implementation, the data structures for parameters and solution vectors are stored locally to each subdomain and interface. Specifically, variables are represented as numpy arrays and parameters as a combination of numpy arrays and dedicated classes.

4.2.1 Discretization classes

For the implementation of discretizations, it is useful to differ between the schemes themselves, their implementation, and the application of a discretization object to a specific grid and parameter set, which produces a discretization matrix. All discretization schemes are implemented as classes which are designed to act on individual subdomains or interfaces. In most cases, there is a one-to-one correspondence between terms in the governing equations and discretization. As an example, the compressible flow equation on a subdomain will be specified by assigning discretizations of the accumulation and diffusion term to a pressure variable, as is shown in Fig. 7.

A compatible discretization class should implement a method for discretization, which computes coefficients that will enter into a discretization matrix. Furthermore, the class needs a method for assembly of matrix and right-hand side. The act of discretization and assembly should together produce a local discretization matrix, usually in the form of a sparse matrix represented using the SciPy library and a right-hand side represented as a numpy array.

There are important differences between discretization classes for subdomains and interfaces: Subdomain discretizations have access only to the subdomain grid and its associated data and assemble a matrix local to the subdomain. An interface discretization is responsible for coupling variables on the neighboring subdomains, and it therefore has access to the relevant subdomain discretizations and data in addition to information local to the interface. Thus, an interface discretization may put additional requirements on a subdomain discretization, see Section 4.2.2 for an example. The assembly method in the interface discretization should treat both the interface equation and the discrete couplings of the interface law to the neighboring subdomains.

In PorePy, subdomain discretization schemes are available for diffusion, advection, and mechanical deformation, as well as mass matrices for accumulation terms. Specifically, diffusion processes can be discretized by the lowest order Raviart-Thomas mixed finite elements combined with a piecewise constant pressure approximation (RT0-P0) [56], the lowest order mixed virtual element method (MVEM) combined with a piecewise constant pressure approximation [57, 58], and by two finite volume schemes: the two- and multipoint flux approximations (TPFA and MPFA, respectively). Advection terms can be discretized by a first-order upstream scheme. Mechanical deformation is discretized by the multipoint stress approximation (MPSA) [59, 60], also extended to poroelasticity [61] and thermo-poroelasticity [62].

On interfaces, discretization schemes in PorePy cover the interface diffusion law (3.3), and an upstream scheme for the advection term (3.9). The discretization of the contact mechanics (Eqs. (3.14) and (3.15)) is implemented by a semi-smooth Newton method to deal with the discontinuities in the solution, for details we refer to [49, 63]. The available discretizations on subdomains and interfaces can also be used as building blocks for more complex problems; for instance, the simulations of thermo-poroelasticity with fracture deformation reported in [64] utilized several of the discretization schemes mentioned above.

In the following, we present the implementation of two examples of combined subdomain and interface discretizations, allowing us to discuss different aspects in the design and implementation of mixed-dimensional problems.

4.2.2 Subdomain coupling for discretization of mixed-dimensional flow

3.1, focusing on the division of responsibilities between subdomain and interface discretizations. The discretization of the interface law (3.3) is implemented in the class RobinInterfaceLaw, which in itself is simple, but has an instructive approach to communication with the adjacent subdomain discretizations. From the model in Section 3.1, we see that for a discretization on a generic subdomain Ω_i to interact with the interface problem, we need to provide operators which:

1) Handle Neumann boundary data on the form $\Xi_j^i \lambda_j$ for all interfaces Γ_j for which Ω_i is the higher-dimensional neighbor.
2) Handle source terms $\Xi_j^i \lambda_j$ from interfaces Γ_j for which Ω_i is the lower-dimensional neighbor.
3) Provide a discrete operator $tr\, p_i$ to be combined with Π_j^i to project the pressure to interfaces Γ_j, $j \in \check{S}_i$.
4) Provide a pressure p_i that can be projected to interfaces Γ_j, $j \in \widehat{S}_i$ using Π_j^i.

RobinInterfaceLaw assumes that the subdomain discretization has dedicated methods, with specified names, that handle each of these four operations. Thus, any discretization class aimed at individual subdomains can be made compatible with RobinInterfaceLaw, and thus applicable to mixed-dimensional problems, provided the four required methods are implemented. Moreover, all of these are readily available in any reasonable implementation of a discretization scheme for elliptic equations. Examples of how RobinInterfaceLaw is set up to interact with subdomain discretizations can be found in Figs. 7 and 10.

It is instructive to write out the structure of the coupled system for our case with two subdomains Ω_h and Ω_l separated by an interface Γ_j. Denote by y_h, y_l, and ξ_j the vectors of discrete unknowns in Ω_h, Ω_l, and on Γ_j, respectively. As we make no assumptions that the same discretization scheme is applied in both subdomains, these may contain different sets of unknowns. The discrete system can then be represented on the generic form

$$\begin{pmatrix} A_h & 0 & N_h\Xi_j^h \\ 0 & A_l & S_l\Xi_j^l \\ -\Pi_j^h P_h & \Pi_j^l P_l & M_j \end{pmatrix} \begin{pmatrix} y_h \\ y_l \\ \xi_j \end{pmatrix} = \begin{pmatrix} f_h \\ f_l \\ 0 \end{pmatrix}. \tag{4.1}$$

Here, A_h and A_l are the fixed-dimensional discretizations on the subdomains and f_h and f_l the corresponding source and sink terms. N_h is the discretization of Neumann boundary conditions on Ω_h, and S_l is the discretization of source terms in Ω_l. Furthermore, P_h provides a discrete representation of the pressure trace operator on Ω_h and P_l gives the pressure unknowns in Ω_l; the latter is an identity operator for the integral formulations presented on primal form and strips away flux unknowns in the dual formulation. Finally, M_j represents the normal permeability term in (3.3) and is discretized directly by RobinCoupling. In accordance with the second constraint on mixed-dimensional modeling discussed in Section 2.2, there is no direct coupling between Ω_h and Ω_l as seen from the 0 entries in the matrix.

The PorePy implementation of the above method represents the mortar variable by piecewise constant functions. Our implementation for the coupled mixed-dimensional problem relies on the analysis carried out in [39], which provides a theoretical background to obtain a stable global scheme with full flexibility in choosing heterogeneous discretization schemes between the subdomains. We also note that the interface discretization for many other classes of equations, such as the advection-diffusion problem presented in Section 3.2, follows a similar approach.

4.2.3 Subdomain couplings for contact mechanics in poroelastic media

As a second example of the matrix structure produced by a subdomain and interface coupling, we consider the model for fracture deformation introduced in Section 3.3. This can be considered a complex model, in that the traction balance on the interface involves multiple variables on Ω_h, Ω_l, and Γ_j. Specifically, the equations for the momentum balance presented in Section 3.3 can be represented in matrix form as

$$\begin{pmatrix} A_h & B_h & D_h\Xi_j^h & 0 & 0 \\ 0 & 0 & U_l\Xi_j^l & 0 & T_l \\ \Pi_j^h T_h & \Pi_j^h G_h & \Pi_j^h S_h \Xi_j^h & -\Pi_j^l G_l & \pm\Pi_j^l \end{pmatrix} \begin{pmatrix} u_h \\ p_h \\ u_j \\ p_l \\ \sigma_l \end{pmatrix} = \begin{pmatrix} b_h \\ r \\ 0 \end{pmatrix}. \tag{4.2}$$

Here, the first row represents the momentum balance with the contribution of the mortar displacement variables on the momentum balance in Ω_h. In practice, this takes the form of a Dirichlet boundary condition discretized as D_h, while A_h, B_h, and b_h represent discretization of poroelasticity in Ω_h. In the second row, the matrices U_l and T_l represent the linearized fracture conditions, i.e., the relation between u_j and σ_l stated in Eqs. (3.14) and (3.15), with contributions from the previous Newton iteration and time step entering in $r = r(u_j, \sigma_l)$. The third row represents Newton's third law over the interfaces, and thus is a discretization of Eq. (3.13). The first three terms provide the traction on the two fracture walls reconstructed from the variables on $\partial_j\Omega_h$ and Γ_j, where S_h represents a mapping from the Dirichlet boundary condition to tractions. The two last terms relate these tractions to the variables in Ω_l, where G_l represents $n_h\alpha_l$, while the $\pm$ in the last term accounts for the fracture side. We emphasize that neither the inter-dimensional contributions to mass conservation nor the coupling for mass conservation is included in (4.2); this is handled by the corresponding internal subdomain discretizations and additional coupling discretizations in the form discussed in Section 4.2.2.

In terms of implementation, the interface equations in (4.2) are in fact split into three different classes: One which handles the interaction between u_h, u_j, and σ_l and two that represent the fluid traction on Γ_j from p_h and p_l, respectively. The most interesting of these classes is the first, termed PrimalContactCoupling, which is used for purely mechanical problems; the discretization of the contact problem that produces the matrices U_l and T_l for the current state of $[\![u_j]\!]$ and σ_l is outsourced to a separate class ColoumbContact. An illustration of how PrimalContactCoupling is set up to interact with the surrounding variables and discretizations is given in the context of Sneddon's problem of fracture deformation (see Fig. 16 in Section 5.3).

4.3 Global assembly of mixed-dimensional multiphysics problems

As discussed in Section 4.2, PorePy requires only specification of variables and discretizations locally on subdomains and interfaces. The global organization is left to the Assembler class, which has the following responsibilities: First, to assign a global numbering of the degrees of freedom of all local variables. Second, to apply all assigned discretization schemes. Third, to assemble the sparse global linear system. The user interface to the Assembler is simple;

numbering of degrees of freedom is handled in the object initialization, while the class has dedicated methods for discretization and assembly. The underlying implementation of these methods is elaborate and involves nested loops over the GridBucket. For global discretization, all local discretization objects are identified, and their respective discretization methods invoked. In the assembly operation, the local discretization matrices are placed in the global linear system according to the degree of freedom of the associated local variable(s).

It is instructive to consider the structure of the global linear system in the setting of a multiphysics problem with more than one primary variable. It has a double block structure, with one set of blocks stemming from the geometric division into subdomains and interfaces. Within each subdomain and interface, there is a second set of blocks, with one block per variable or variable pair (for off-diagonal blocks). This information, which is useful for design of tailored preconditioners and linear solvers as well as post-processing and visualization, can be accessed through the Assembler. We emphasize that the implementation of the Assembler is general in the sense that it can be applied to new discretizations and governing equations without modification.

The bottom-up approach to the assembly of variables and discretizations to some degree favors flexibility over computational speed. The overhead in construction and manipulation of matrices, independent of matrix size and separate from the cost of discretization, is minor but can become notable when repeated many times, e.g., in time-dependent and non-linear problems. For problems with many subdomains, the cost in using local assembly can become prohibitively high. Specifically, the cost has been pronounced in simulations of non-linearly coupled flow and transport, as reported in [65] and also in Section 6.1. As a remedy, which is also compatible with the automatic differentiation (AD) module in PorePy, the Assembler also provides methods to construct global discrete operators.

4.4 Solvers and visualization

PorePy has no native support for linear solvers, but instead relies on external libraries for solving linear systems. The structure of the linear systems obtained for mixed-dimensional is non-standard compared with that of similar fixed-dimensional problems. Thus, if the linear system is to be solved by iterative methods, traditional preconditioners cannot be expected to perform well, and specialized methods may be preferable. Preconditioners for mixed-dimensional problems are an immature research field, see however [66, 67] for examples on how PorePy can be combined with dedicated solvers for mixed-dimensional problems.

Finally, visualization is handled by an export filter to the vtk/vtu format, which can be read for instance by Paraview [68]. To aid analysis of simulation results, the export preserves the link between the data and its associated dimensions.

5 Validation

In this section, we validate our modeling framework and its implementation in PorePy by probing discretization schemes, multiphysics problems, and time-dependent problems through three test cases: a benchmark for flow problems in 2d fractured media, Mandel's problem for poroelasticity, and Sneddon's problem for fracture deformation in elastic media. The cases thus supplement previous testing of PorePy, reported in [38, 69–71]. The supplementary material provides detailed setups, including parameters, for all simulations in Sections 5 and 6. Scripts that reproduce all results reported herein can be accessed at [72], see that reference or the supplementary material for installation instructions.

5.1 Flow in 2d fractured porous media

To validate the mixed-dimensional flow discretization, we consider Benchmark 3 of [73], which describes the incompressible single-phase flow problem in a fractured domain presented in Section 3.1. The fracture network contains intersecting and isolated fractures (see Fig. 9). The network contains both highly conductive and blocking fractures, see the supplementary material for parameter details.

The aim of this case is twofold — we benchmark our code against well-established methods in the literature and illustrate PorePy's flexibility in assigning heterogeneous subdomain discretizations. We consider four groups of discretization schemes and simulation grids: first, three homogeneous (the same for all the subdomains) discretizations: TPFA, MPFA, and RT0-P0. Second, a case with the MVEM, where the cells of the rock matrix are constructed by a clustering procedure starting from a more refined simplicial grid, see [70] for details. Third, two heterogeneous discretizations where RT0-P0 and MVEM for the rock matrix are combined with TPFA for the fractures. Fourth, a case where the fracture grid is twice as fine as the matrix grid, with the mortar grids non-conforming to the surrounding grids (labeled Non-Matching) discretized using the RT0-P0 scheme. We use simplex grids in all cases that do not involve MVEM. A code snippet that highlights the assignment of heterogeneous discretizations is given in Fig. 10.

Figure 9 shows the domain with fractures, boundary conditions, and a representative numerical solution. The figure also depicts a plot of the pressure along the line $(0, 0.5) - (1, 0.9)$. We observe good agreement between the solutions obtained in PorePy and the reference solution of [73], which is a solution of the equi-dimensional problem computed on a very fine grid. We also perform a refinement study using a sequence of three grids to compute the error relative to the

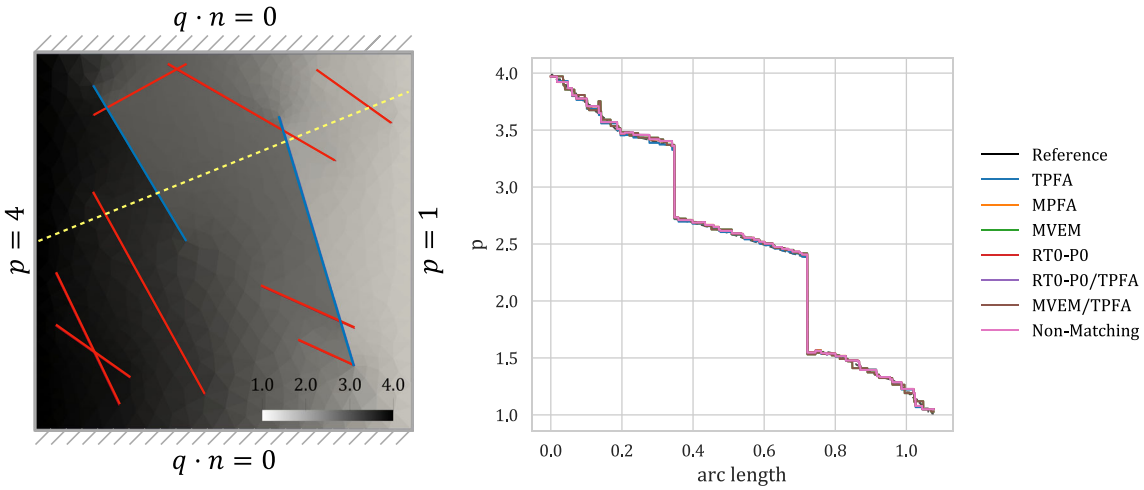

Fig. 9 Left: A solution obtained with MPFA on the coarsest grid showing the fracture network and the problem setup. The red lines represent conductive fractures whereas the blue lines are blocking fractures. The yellow line indicates the line of the pressure profile. Right: Pressure profiles for the discretization schemes used in the validation

reference solution, as done in the original benchmark. Figure 11 shows the decay of the normalized L^2 error for the rock matrix and the union of the fracture subdomains. In the former, we notice a first order of convergence for all the considered methods. The convergence rate for the fracture subdomains is sublinear, as was also observed in the original benchmark.

5.2 Mandel's problem in poroelasticity

The next test case considers a poroelastic material, with a setup defined by Mandel's problem [74, 75], for which an analytical solution is available. While the problem geometry does not include lower-dimensional objects, the case tests the implementation of the poroelastic code and shows the framework's flexibility to

```
matrix_discr = pp.RT0('flow')     # Discretization in the matrix
fracture_discr = pp.Tpfa('flow')  # Discretization in fractures and intersections

for g, d in gb:
    if g.dim == 2:  # This is the matrix grid
        # RT0 has both cell and face unknowns
        d[pp.PRIMARY_VARIABLES] = {'p': {'cells': 1, 'faces': 1}}
        d[pp.DISCRETIZATION] = {'p': {'diffusion': matrix_discr}}
    else:  # Fracture or fracture intersection grid.
        # TPFA has only cell unknowns
        d[pp.PRIMARY_VARIABLES] = {'p': {'cells': 1}}
        d[pp.DISCRETIZATION] = {'p': {'diffusion': fracture_discr}}

for e, d in gb.edges():  # Loop over all interfaces (e) and their associated data (d)
    g_l, g_h = gb.nodes_of_edge(e)  # Get grids of neighboring subdomains
    d[pp.PRIMARY_VARIABLES] = {'mortar_flux': {'cells': 1}}  # Primary variable

    if g_h.dim == 2:  # Adapt interface discretization to neighboring subdomains
        interface_discr = pp.RobinCoupling('flow', matrix_discr, fracture_discr)
    else:  # Omega_l is an intersection point
        interface_discr = pp.RobinCoupling('flow', fracture_discr, fracture_discr)

    # Define coupling
    d[pp.COUPLING_DISCRETIZATION] = { 'interface_flux':  # Identifier of this term
        {g_h: ('p', 'diffusion'),
         g_l: ('p', 'diffusion'),
         e: ('mortar_flux', interface_discr)}}
```

Fig. 10 Code snippet of the discretization assignment for the combination of RT0-P0 and TPFA. The code can be used as a partial replacement of the green section in Fig. 7. Note that the parameter definition is not included in the snippet

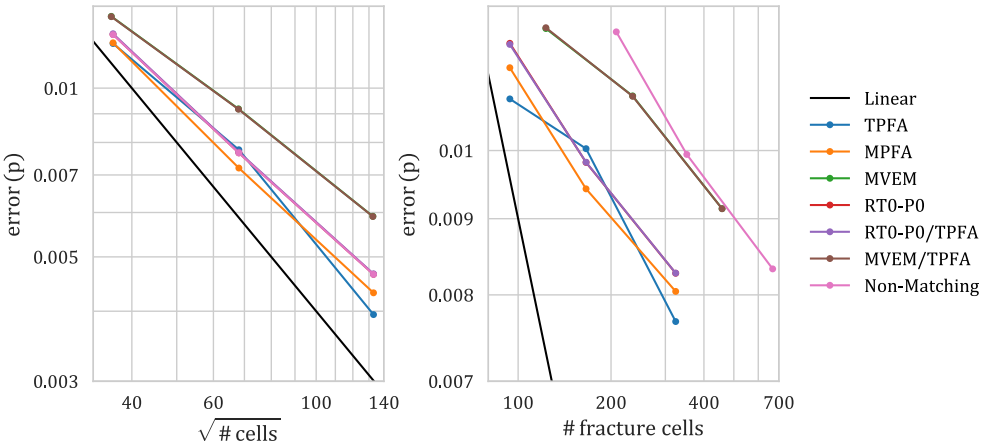

Fig. 11 Left: Convergence of the pressure unknown for the matrix subdomain for the simulations reported in Section 5.1. Right: Convergence for the pressure unknown for the fracture subdomains

deal with coupled problems and time-dependent mixed boundary conditions. The original problem consists of an isotropic poroelastic slab of width $2a$ and height $2b$ sandwiched by two rigid plates (Fig. 12). Initially, two compressive constant loads of intensity $2F$ are applied to the slab at $y = \pm b$. At $x = \pm a$, fluid is free to drain, and edges are stress free. Gravity contributions are neglected.

The problem is modeled using the quasi-static Biot equations, as presented in Section 3.3. Exploiting the symmetry of the problem, we focus on the positive quarter domain Ω', rather than the full domain Ω, see Fig. 12 for an illustration and for boundary conditions. Note that the vertical displacement at the top of the domain is time-dependent and given by the exact solution, see [76].

The simulation parameters were taken from [77], see also the supplementary material for details. The coupled problem is discretized in space using MPSA and MPFA for the mechanics and flow, respectively. For the time discretization, we use implicit Euler. The computational grid is unstructured and composed of 622 triangular elements. The results are shown in Fig. 13 in terms of dimensionless quantities and are in good agreement with [77] for both pressure and displacement.

In Fig. 14, we show a code snippet illustrating the assembly of a generic poroelastic problem using MPSA/MPFA in PorePy. One primary variable for each subproblem must be specified, namely displacement for the mechanics (variable 0) and pressure for the flow (variable 1). There are five terms (plus one stabilization term) involved in the discretization of the Biot equations. We label them with subscripts kl identifying the impact on variable k from variable l. The numbering also corresponds to the placement in the 2×2 block discretization matrix, with the first row representing the momentum balance and the second row the mass balance.

The Mpsa class is used to obtain the divergence of the stress (term_00), which corresponds to the first diagonal block. For the second diagonal block, term_11_0 and term_11_1 refer to the discretization of the fluid accumulation and fluid flux (after applying implicit Euler) obtained using the classes ImplicitMassMatrix and ImplicitMpfa, respectively. In addition, term_11_2 is a stabilization term arising naturally from the discretization process [61].

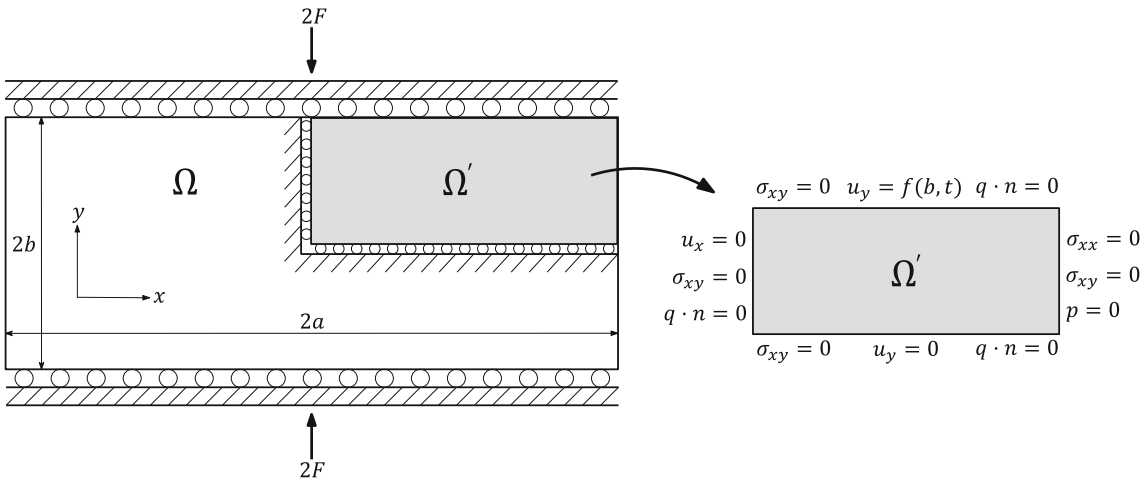

Fig. 12 Mandel's problem. Left: Schematic representation of the full and positive quarter domains, Ω and Ω'. Right: Quarter domain showing the boundary conditions

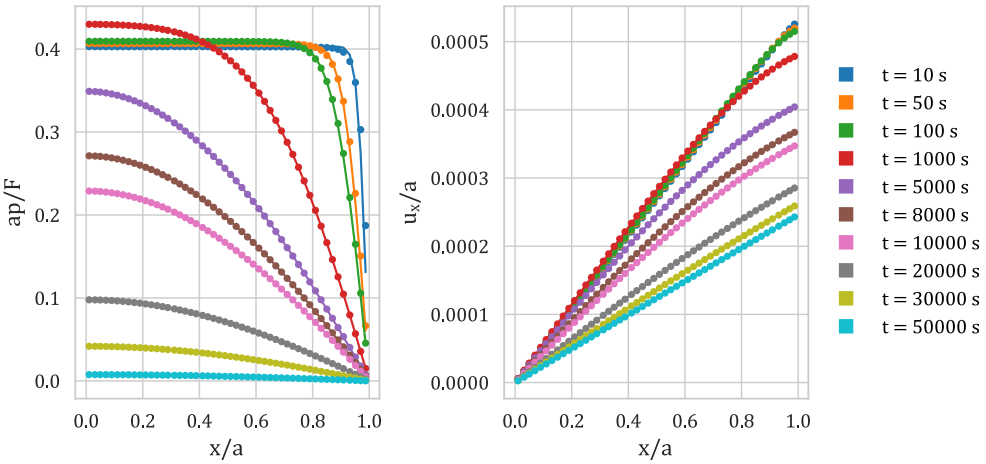

Fig. 13 Analytical (solid lines) and MPSA/MPFA (dots) solutions to Mandel's problem. The dimensionless profiles for the pressure (left) and the horizontal displacement (right) are shown for several times

Lastly, term_01 and term_10 are the off-diagonal coupling blocks representing respectively the terms involving the pressure gradient (obtained with GradP) and the divergence of the displacement field (obtained with DivU).

5.3 Sneddon's problem of fracture deformation

In this example, a square domain with a single fracture located in the middle is considered. The fracture forms an angle β with the horizontal direction (see Fig. 15) and is subjected to a constant pressure p_0, which can be interpreted as a pair of normal forces acting on either side of the fracture. An analytical solution for the relative normal displacement along the fracture was derived by Sneddon [78] for an infinite domain, and has the following form:

$$[\![u_j]\!]_n(d_f) = \frac{(1-\nu)p_0 L}{G}\sqrt{1-\frac{d_f^2}{\left(\frac{L}{2}\right)^2}}, \tag{5.1}$$

```
## Primary variables for poroelasticity problems
v_0 = 'u'  # displacement
v_1 = 'p'  # pressure

## Name of the five terms of the equation + additional stabilization term
term_00 = 'stress_divergence'       # div symmetric grad u
term_11_0 = 'fluid_mass'            # d/dt beta p
term_11_1 = 'fluid_flux'            # div (rho g - K grad p)
term_11_2 = 'stabilization'
term_01 = 'pressure_gradient'       # alpha grad p
term_10 = 'displacement_divergence' # d/dt alpha div u

## Store in the data dictionary and specify discretization objects
d[pp.PRIMARY_VARIABLES] = {v_0: {'cells': g.dim}, v_1: {'cells': 1}}
d[pp.DISCRETIZATION] = {
    # Momentum balance equation
    v_0: {term_00: pp.Mpsa(kw_m)},
    # Mass conservation equation
    v_1: {
        term_11_0: pp.ImplicitMassMatrix(kw_f, v_1),
        term_11_1: pp.ImplicitMpfa(kw_f),
        term_11_2: pp.BiotStabilization(kw_f, v_1),
    },
    # Pressure contribution to the momentum balance equation
    v_0 + '_' + v_1: {term_01: pp.GradP(kw_m)},
    # Displacement contribution to the mass conservation equation
    v_1 + '_' + v_0: {term_10: pp.DivU(kw_m, kw_f, v_0)},
}
```

Fig. 14 Code snippet illustrating the terms involved in the assembly of a poroelastic problem using MPSA/MPFA in PorePy. The snippet highlights assignment of discretizations for multiphysics problems within a subdomain

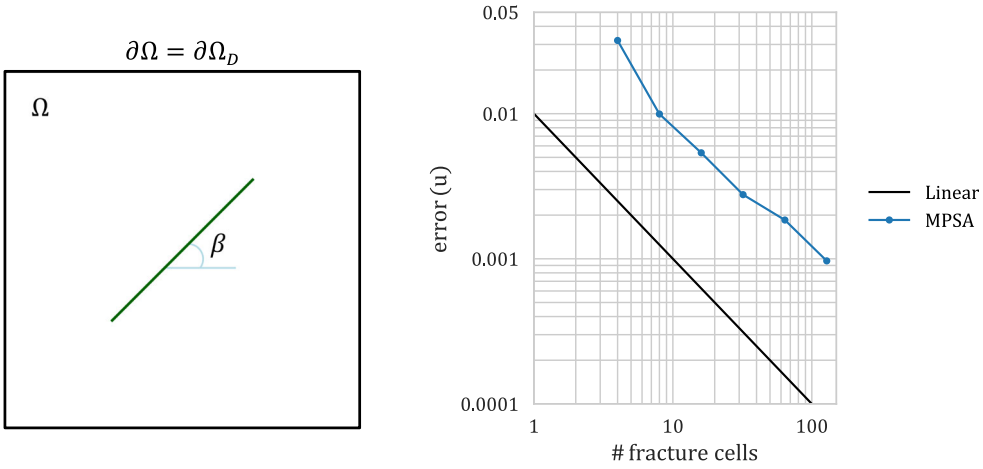

Fig. 15 Setup and convergence of Sneddon's problem. Left: Schematic representation of the domain. Right: Average convergence behavior of the relative normal displacement along the fracture. Each dot corresponds to the average of 140 simulations

where ν and G are the Poisson's ratio and shear modulus, respectively, L is the fracture length, and d_f denotes the distance from the center of the fracture.

In our calculations, the condition of infinite domain is replaced with a Dirichlet boundary, where the prescribed displacement is set equal to the analytical solution calculated using the procedure illustrated in [79]. The accuracy of the numerical solution is very sensitive to the discretization, specifically the cell configuration at the fracture tips [46]. To reduce the dependency on specific grid realizations, the values of the numerical solution reported in Fig. 16 are the average of a group of $20 \times 7 = 140$ computations per level of grid resolution, with 7 different fracture angles β in the range 0°–30° and 20 grid realizations per fracture. With six levels of grid refinement, the full study contains $20 \times 7 \times 6 = 840$ simulations. Figure 16 summarizes the results in the form of the error in relative normal displacement between the analytical solution (5.1) and the numerical solution as a function of the fracture resolution, i.e., number of fracture elements. The method provides first-order convergence on average.

Finally, the code snippet in Fig. 16 indicates the key parts of the variable and discretization assignment for the contact mechanics problem. The classes to note are ColoumbContact, which represents Eqs. (3.14) and (3.15), and the interface discretization PrimalContactCoupling, see also the discussion in Section 4.2.3.

6 Applications: multiphysics simulations

Having established the accuracy of PorePy for central test cases that involve mixed-dimensional geometries, we proceed to present two multiphysics cases of high application relevance: A non-linearly coupled flow and transport problem, and fracture reactivation caused by fluid injection. The motivation for the simulations is to illustrate further capabilities of the modeling framework and its PorePy implementation, including simulations on complex 3d fracture networks, automatic differentiation applied to non-linear problems, non-matching grids, and simulation of fracture deformation in a poroelastic setting.

6.1 Fully coupled flow and transport

We consider the injection of a more viscous fluid into a domain initially filled with a less viscous fluid. The two fluids are miscible and have equal densities; thus, they can be modeled as two components in a single-phase system, as described in Section 3.2. The viscosity of the mixture of fluids given by $\mu_i(c_i) = \exp(c_i)$, for the mass fraction $c_i \in [0, 1]$, which is 0 if only the less viscous fluid is present and 1 if only the more viscous fluid is present. In the parameter regime studied in this example, the transport in the fractures is advection dominated, while the transport in the rock matrix is dominated by diffusion, see the supplementary material for details.

The time derivative is approximated using an implicit Euler method, which gives a fully implicit scheme for the primary variables pressure and mass fraction. The spatial terms are discretized by a finite volume method, with simple upstream for advective terms, and TPFA for fluxes and diffusive terms. We apply forward automatic differentiation implemented in PorePy to obtain the Jacobian of the global system of equations, which is then used in a standard Newton method to solve the non-linear problem. The convergence criterion is given by the maximum norm of the residual vector with a tolerance 10^{-9}.

The mixed-dimensional domain considered in this example consists of one 3d domain, 15 2d fracture domains, 62 1d domains, and 9 0d domains. On this geometry, two computational grids are constructed: The first has matching grids in all dimensions, with in total 20,812 cells, out of which 16,766 are 3d cells and 3,850 are 2d fracture cells. The second mixed-dimensional grid has a 3d grid identical to the first grid, whereas the lower-dimensional objects are assigned refined grids with in total 13,839 2d fracture cells; thus, the 3d-2d interfaces have non-matching grids. The combination of the

```
Nd = gb.dim_max()  # Get ambient dimension
# For the 2d domain we solve linear elasticity with MPSA
mpsa = pp.Mpsa('mechanics')

# The assembler expects that all variables are assigned a discretization. Create a
# void discretization object; the friction discretization is set via the interface law
empty_discr = pp.VoidDiscretization('friction', Nd)

# Define discretization parameters
for g, d in gb:
    if g.dim == Nd:  # Omega_i is the matrix
        d[pp.PRIMARY_VARIABLES] = {'u': {'cells': Nd}}
        d[pp.DISCRETIZATION] = {'u': {'mpsa': mpsa}}
    else:  # Omega_i is a fracture
        d[pp.PRIMARY_VARIABLES] = {'contact': {'cells': Nd}}
        d[pp.DISCRETIZATION] = {'contact': {'empty': empty_discr}}

# For the 1d domain we define a contact condition
coloumb = pp.ColoumbContact('friction', Nd)
# Define a contact condition on the mortar grid
contact = pp.PrimalContactCoupling('friction', mpsa, coloumb)

# Loop over all interfaces (e) and their associated data (d)
for e, d in gb.edges():
    g_l, g_h = gb.nodes_of_edge(e)

    d[pp.PRIMARY_VARIABLES] = {'interface_u': {'cells': Nd}}
    d[pp.COUPLING_DISCRETIZATION] = {'friction': {
        g_h: ('u', 'mpsa'),
        g_l: ('contact', 'empty'),
        (g_h, g_l): ('interface_u', contact)}}
```

Fig. 16 Code snippet that illustrates variable and discretization assignment for Sneddon's problem, discretized using the contact mechanics functionality in PorePy. The code can be used as a partial replacement of the green section in Fig. 7

non-linearity and the non-matching grids provides a challenging test for the robustness of the PorePy implementation of subdomain couplings and provides an illustration of the framework's flexibility.

Figure 17 shows the average mass fraction profile in the fractures for the two grids. There are no significant differences between the two cases, indicating the stability of the implementation of the non-matching case. Figure 18 shows a snapshot of the mass fraction in the fractures and the rock matrix at time $t = 20$. The diffusive front in the rock matrix has only moved a few grid cells at the break-through; however, due to the diffusion and advection from the fractures to the rock matrix, the mass fraction has increased in considerable parts of the rock matrix. We observe no irregularities for the solution produced on the non-matching grid in this case, suggesting PorePy's ability to deal with non-standard grid couplings also for challenging physical regimes.

6.2 Poroelasticity and fracture deformation

The final example aims at demonstrating the modeling framework's and PorePy's applicability to non-standard combinations of physical processes in different domains and thereby its potential for method development and prototyping. With the critical events taking place on individual fractures as a result of processes in the rock matrix, it also serves as an example of the importance of incorporating dynamics of both the matrix and explicitly represented fractures, as done in DFM models.

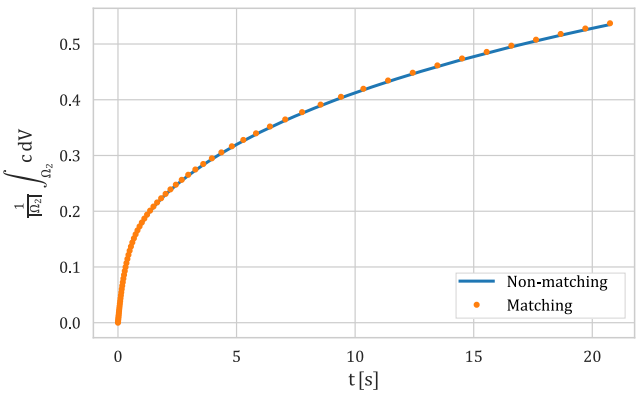

Fig. 17 Fully coupled flow and transport: Comparison of average mass fraction in the fracture network for a simulation with matching grids and a simulation with non-matching grids

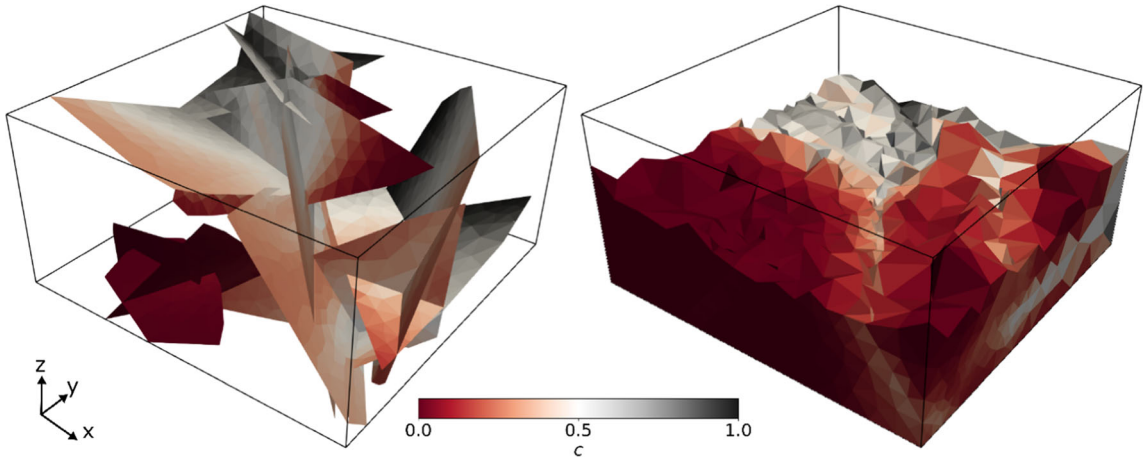

Fig. 18 Fully coupled flow and transport: Mass fraction in the fractures (left) and in the rock matrix (right) for the coupled flow and transport problem given in Section 3.2 at the end time of the simulation ($t = 20$). In the right figure, the rock matrix domain is cropped, and the fractures removed to reveal the mass fraction inside the domain. The black lines indicate the domain boundary. Non-matching grids are used with the fracture grids being much finer than the grid in the rock matrix

Specifically, we consider the model equations for coupled poroelasticity and fracture deformation presented in Section 3.3. The poroelastic deformation of the host rock is discretized with MPSA, while the fluid flow in the fractures is discretized with MPFA. The discretization of the contact mechanics follows the structure outlined in Section 4.2.3, and temporal discretization is performed using implicit Euler.

We consider a reservoir of idealized geometry containing three non-intersecting fractures numbered from 1 through 3, whereof the first contains an injection well (see Fig. 19). On this geometry, we solve the governing equations presented in Section 3.3. We impose injection over a 25-day period and an anisotropic background stress regime, producing a scenario well suited to demonstrate different fracture dynamics. We investigate the dynamics both during the injection phase and during the subsequent 25-day relaxation phase, at the end of which the pressure has almost reached equilibrium once more. The full set of parameters may be found in the supplementary material.

The dynamics on the fractures throughout the simulation are summarized in Fig. 19, while the spatial distribution of the fracture displacement jumps at the end of the injection phase is shown in Fig. 20. The figures show how the simulation captures the complex dynamics both during and after injection, and thus highlight how the explicit fracture representation allows for detailed studies of fracture deformation.

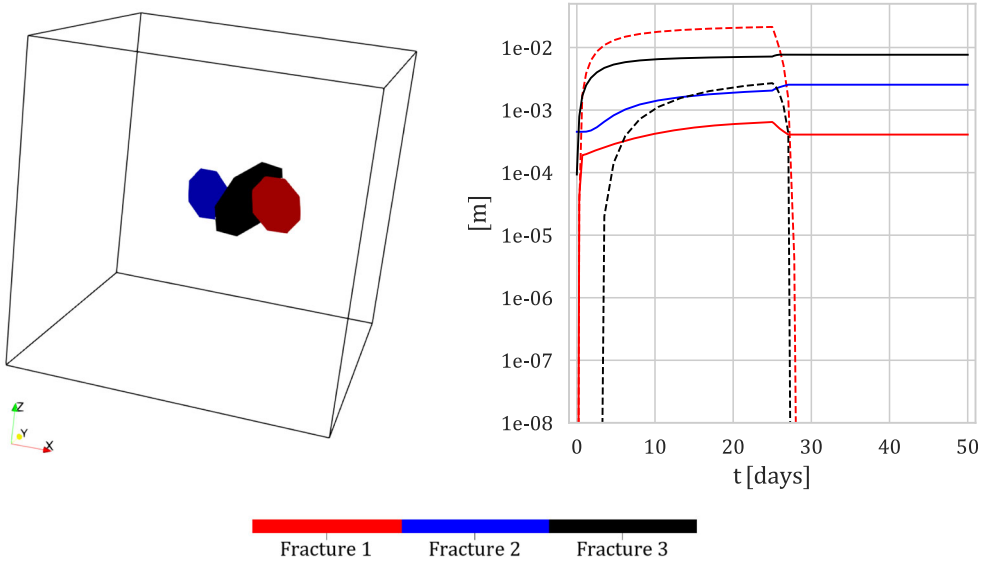

Fig. 19 Left: Domain geometry with numbering of the three fractures. Fluid is injected in fracture 1 during the first 25 days, after which the well is shut. Right: L^2-norm normalized by fracture area of the normal (dashed lines) and tangential (solid lines) displacement jumps for each fracture

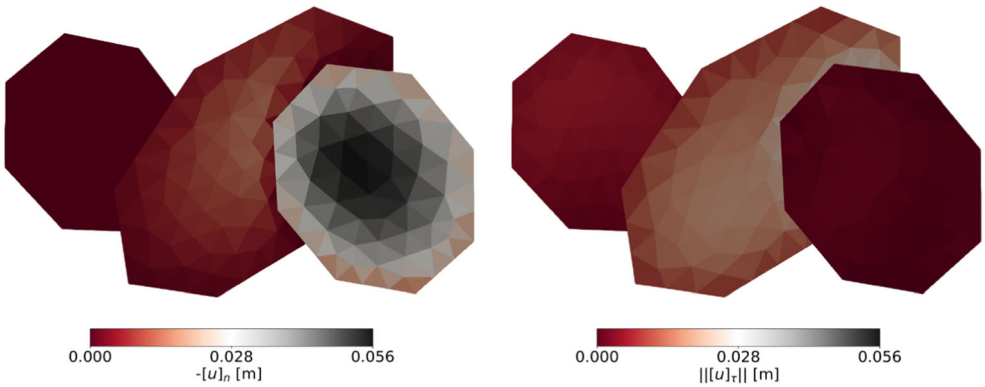

Fig. 20 Normal and tangential displacements jump on the fractures at the end of the injection phase to the left and right, respectively. The orientation of the fracture network corresponds to that in Fig. 12, with the injection fracture to the right

7 Conclusions

The complexity in modeling and simulation of multiphysics processes in fractured porous media, combined with a strong current research focus and corresponding developments, calls for flexible simulation tools that facilitate rapid prototyping of models and discretization methods. This paper presents design principles for such software together with their implementation in the open-source simulation tool PorePy. The combined framework for modeling and simulation is based on the discrete fracture matrix model, where fractures and their intersections are represented as separate lower-dimensional geometric objects. The framework facilitates flexibility for multiphysics dynamics and reuse of existing code written for non-fractured domains; hence, it is well suited for extending other software packages to mixed-dimensional problems.

The open-source software PorePy demonstrates the capabilities of the suggested framework: It provides automatic gridding of complex fracture networks in two and three dimensions, and contains implemented numerical methods for flow, transport, poroelastic deformation of the rock, and fracture deformation modeled by contact mechanics. The implementation performs well for benchmark problems in flow, poroelastic deformation, and fracture deformation. Furthermore, multiphysics simulations of fully coupled flow and non-linear transport and of fracture deformation under poroelastic deformation of a domain demonstrate the versatility of the software.

Acknowledgments The authors thank two anonymous reviewers for the comments and suggestions that helped to improve the quality of the paper.

Funding Open Access funding provided by University of Bergen. This work has been funded in part by Norwegian Research Council grant 250223, 244129/E20, 267908/E20, and 274883, and by a VISTA Scholarship from the Norwegian Academy of Science and Letters.

References

1. Berkowitz, B.: Characterizing flow and transport in fractured geological media: a review. Adv. Water Resour. **25**(8–12), 861–884 (2002). https://doi.org/10.1016/S0309-1708(02)00042-8
2. Martin, V., Jaffré, J., Roberts, J.E.: Modeling fractures and barriers as interfaces for flow in porous media. SIAM J. Sci. Comput. **26**(5), 1667–1691 (2005). https://doi.org/10.1137/S1064827503429363
3. Barton, N., Bandis, S., Bakhtar, K.: Strength, deformation and conductivity coupling of rock joints. Int. J. Rock Mech. Min. Sci. Geomech. Abstr. **22**(3), 121–140 (1985). https://doi.org/10.1016/0148-9062(85)93227-9
4. Frih, N., Roberts, J.E., Saada, A.: Modeling fractures as interfaces: a model for Forchheimer fractures. Comput. Geosci. **12**(1), 91–104 (2008). https://doi.org/10.1007/s10596-007-9062-x
5. Rutqvist, J., Wu, Y.-S., Tsang, C.-F., Bodvarsson, G.: A modeling approach for analysis of coupled multiphase fluid flow, heat transfer, and deformation in fractured porous rock. Int. J. Rock Mech. Min. Sci. **39**(4), 429–442 (2002). https://doi.org/10.1016/S1365-1609(02)00022-9
6. Burnell, J., et al.: Geothermal supermodels: the next generation of integrated geophysical, chemical and flow simulation modelling tools. Proc World Geotherm. Congr. 7 (2015)
7. Pruess, K.: TOUGH2: a general numerical simulator for multiphase fluid and heat flow. Report LBL-29400 (1991)
8. Hammond, G.E., Lichtner, P.C., Mills, R.T.: Evaluating the performance of parallel subsurface simulators: an illustrative example with PFLOTRAN: evaluating the parallel performance of Pflotran. Water Resour. Res. **50**(1), 208–228 (2014). https://doi.org/10.1002/2012WR013483
9. Barenblatt, G.I., Zheltov, I.P., Kochina, I.N.: Basic concepts in the theory of seepage of homogeneous liquids in fissured rocks [strata]. J. Appl. Math. Mech. **24**(5), 1286–1303 (1960). https://doi.org/10.1016/0021-8928(60)90107-6

10. Arbogast, T., Douglas Jr., J., Hornung, U.: Derivation of the double porosity model of single phase flow via homogenization theory. SIAM J. Math. Anal. **21**(4), 823–836 (1990). https://doi.org/10.1137/0521046
11. Lemonnier, P., Bourbiaux, B.: Simulation of naturally fractured reservoirs. State of the art: part 1 – physical mechanisms and simulator formulation. Oil Gas Sci. Technol. Rev. L'Institut Fr. Pétrole. **65**(2), 239–262 (2010). https://doi.org/10.2516/ogst/2009066
12. Lemonnier, P., Bourbiaux, B.: Simulation of naturally fractured reservoirs. State of the art: part 2 – matrix-fracture transfers and typical features of numerical studies. Oil Gas Sci. Technol. – Rev. L'Institut Fr. Pétrole. **65**(2), 263–286 (2010). https://doi.org/10.2516/ogst/2009067
13. Hyman, J.D., Karra, S., Makedonska, N., Gable, C.W., Painter, S.L., Viswanathan, H.S.: dfnWorks: a discrete fracture network framework for modeling subsurface flow and transport. Comput. Geosci. **84**, 10–19 (2015). https://doi.org/10.1016/j.cageo.2015.08.001
14. Erhel, J., de Dreuzy, J.-R., Poirriez, B.: Flow simulation in three-dimensional discrete fracture networks. SIAM J. Sci. Comput. **31**(4), 2688–2705 (2009). https://doi.org/10.1137/080729244
15. Berrone, S., Pieraccini, S., Scialò, S.: On simulations of discrete fracture network flows with an optimization-based extended finite element method. SIAM J. Sci. Comput. **35**(2), A908–A935 (2013). https://doi.org/10.1137/120882883
16. Berre, I., Doster, F., Keilegavlen, E.: Flow in fractured porous media: a review of conceptual models and discretization approaches. Transp. Porous Media. **130**, 215–236 (2018). https://doi.org/10.1007/s11242-018-1171-6
17. Noorishad, J., Mehran, M.: An upstream finite element method for solution of transient transport equation in fractured porous media. Water Resour. Res. **18**(3), 588–596 (1982). https://doi.org/10.1029/WR018i003p00588
18. Baca, R.G., Arnett, R.C., Langford, D.W.: Modelling fluid flow in fractured-porous rock masses by finite-element techniques. Int. J. Numer. Methods Fluids. **4**(4), 337–348 (1984). https://doi.org/10.1002/fld.1650040404
19. Reichenberger, V., Jakobs, H., Bastian, P., Helmig, R.: A mixed-dimensional finite volume method for two-phase flow in fractured porous media. Adv. Water Resour. **29**(7), 1020–1036 (2006). https://doi.org/10.1016/j.advwatres.2005.09.001
20. Li, L., Lee, S.H.: Efficient field-scale simulation of black oil in a naturally fractured reservoir through discrete fracture networks and homogenized media. SPE Reserv. Eval. Eng. **11**(04), 750–758 (2008). https://doi.org/10.2118/103901-PA
21. Fumagalli, A., Scotti, A.: A reduced model for flow and transport in fractured porous media with non-matching grids. In: Cangiani, A., Davidchack, R.L., Georgoulis, E., Gorban, A.N., Levesley, J., Tretyakov, M.V. (eds.) Numerical Mathematics and Advanced Applications 2011, pp. 499–507. Springer, Berlin (2013)
22. Flemisch, B., Fumagalli, A., Scotti, A.: A review of the XFEM-based approximation of flow in fractured porous media. In: Ventura, G., Benvenuti, E. (eds.) Advances in Discretization Methods, vol. 12, pp. 47–76. Springer International Publishing, Cham (2016)
23. Schwenck, N., Flemisch, B., Helmig, R., Wohlmuth, B.I.: Dimensionally reduced flow models in fractured porous media: crossings and boundaries. Comput. Geosci. **19**(6), 1219–1230 (2015). https://doi.org/10.1007/s10596-015-9536-1
24. Jiang, J., Younis, R.M.: An improved projection-based embedded discrete fracture model (pEDFM) for multiphase flow in fractured reservoirs. Adv. Water Resour. **109**, 267–289 (2017). https://doi.org/10.1016/j.advwatres.2017.09.017
25. Flemisch, B., Darcis, M., Erbertseder, K., Faigle, B., Lauser, A., Mosthaf, K., Müthing, S., Nuske, P., Tatomir, A., Wolff, M., Helmig, R.: DuMux: DUNE for multi-{phase,component,scale, physics,…} flow and transport in porous media. Adv. Water Resour. **34**(9), 1102–1112 (2011). https://doi.org/10.1016/j.advwatres.2011.03.007
26. Matthäi, S.K., Geiger, S., Roberts, S.G., Paluszny, A., Belayneh, M., Burri, A., Mezentsev, A., Lu, H., Coumou, D., Driesner, T., Heinrich, C.A.: Numerical simulation of multi-phase fluid flow in structurally complex reservoirs. Geol. Soc. Lond. Spec. Publ. **292**(1), 405–429 (2007). https://doi.org/10.1144/SP292.22
27. Gaston, D., Newman, C., Hansen, G., Lebrun-Grandié, D.: MOOSE: a parallel computational framework for coupled systems of nonlinear equations. Nucl. Eng. Des. **239**(10), 1768–1778 (2009). https://doi.org/10.1016/j.nucengdes.2009.05.021
28. Breede, K., Dzebisashvili, K., Liu, X., Falcone, G.: A systematic review of enhanced (or engineered) geothermal systems: past, present and future. Geotherm. Energy. **1**(1), 4 (2013). https://doi.org/10.1186/2195-9706-1-4
29. Wang, W., Kolditz, O.: Object-oriented finite element analysis of thermo-hydro-mechanical (THM) problems in porous media. Int. J. Numer. Methods Eng. **69**(1), 162–201 (2007). https://doi.org/10.1002/nme.1770
30. Březina, J., Stebel, J.: Analysis of model error for a continuum-fracture model of porous media flow. In: Kozubek, T., Blaheta, R., Šístek, J., Rozložník, M., Čermák, M. (eds.) High Performance Computing in Science and Engineering, vol. 9611, pp. 152–160. Springer International Publishing, Cham (2016)
31. Lie, K.-A.: An Introduction to Reservoir Simulation Using MATLAB/GNU Octave: User Guide for the MATLAB Reservoir Simulation Toolbox (MRST), 1st edn. Cambridge University Press (2019)
32. Lie, K.-A., Krogstad, S., Ligaarden, I.S., Natvig, J.R., Nilsen, H.M., Skaflestad, B.: Open-source MATLAB implementation of consistent discretisations on complex grids. Comput. Geosci. **16**(2), 297–322 (2012). https://doi.org/10.1007/s10596-011-9244-4
33. Alnæs, M., et al.: The FEniCS Project Version 1.5. Arch. Numer. Softw. **3**, (2015). https://doi.org/10.11588/ans.2015.100.20553
34. Blatt, M., et al.: The distributed and unified numerics environment, Version 2.4. Arch. Numer. Softw. **4**, (2016). https://doi.org/10.11588/ans.2016.100.26526
35. Rathgeber, F., Ham, D.A., Mitchell, L., Lange, M., Luporini, F., Mcrae, A.T.T., Bercea, G.T., Markall, G.R., Kelly, P.H.J.: Firedrake: automating the finite element method by composing abstractions. ACM Trans. Math. Softw. **43**(3), 1–27 (2016). https://doi.org/10.1145/2998441
36. Boon, W.M., Nordbotten, J.M., Vatne, J.E.: Functional analysis and exterior calculus on mixed-dimensional geometries. Ann. Mat. (2020). https://doi.org/10.1007/s10231-020-01013-1
37. Boon, W.M., Nordbotten, J.M.: Stable mixed finite elements for linear elasticity with thin inclusions. arXiv. **1903.01757**, (2019)
38. Nordbotten, J.M., Boon, W.M., Fumagalli, A., Keilegavlen, E.: Unified approach to discretization of flow in fractured porous media. Comput. Geosci. **23**(2), 225–237 (2019). https://doi.org/10.1007/s10596-018-9778-9
39. Geuzaine, C., Remacle, J.-F.: Gmsh: a 3-D finite element mesh generator with built-in pre- and post-processing facilities. Int. J. Numer. Methods Eng. **79**(11), 1309–1331 (2009). https://doi.org/10.1002/nme.2579
40. Karimi-Fard, M.: An efficient discrete-fracture model applicable for general-purpose reservoir simulators. SPE J. **9**(2), (2004). https://doi.org/10.2118/88812-PA
41. Hui, M.-H., Mallison, B., Lim, K.-T.: An innovative workflow to model fractures in a giant carbonate reservoir. Proc. Int. Pet. Tech. Conf. 15 (2008)
42. Berre, I., et al.: Verification benchmarks for single-phase flow in three-dimensional fractured porous media. arXiv. **2002.07005**, (2020)
43. Boon, W.M., Nordbotten, J.M., Yotov, I.: Robust discretization of flow in fractured porous media. SIAM J. Numer. Anal. **56**(4), 2203–2233 (2018). https://doi.org/10.1137/17M1139102

44. Quarteroni, A., Valli, A.: Numerical approximation of partial differential equations, 2nd edn. Springer, Berlin (1997)
45. Garipov, T.T., Karimi-Fard, M., Tchelepi, H.A.: Discrete fracture model for coupled flow and geomechanics. Comput. Geosci. **20**(1), 149–160 (2016). https://doi.org/10.1007/s10596-015-9554-z
46. Ucar, E., Keilegavlen, E., Berre, I., Nordbotten, J.M.: A finite-volume discretization for deformation of fractured media. Comput. Geosci. **22**(4), 993–1007 (2018). https://doi.org/10.1007/s10596-018-9734-8
47. McClure, M.W., Horne, R.N.: An investigation of stimulation mechanisms in Enhanced Geothermal Systems. Int. J. Rock Mech. Min. Sci. **72**, 242–260 (2014). https://doi.org/10.1016/j.ijrmms.2014.07.011
48. Coussy, O.: Poromechanics. Chichester, Wiley (2003)
49. Berge, R.L., Berre, I., Keilegavlen, E., Nordbotten, J.M., Wohlmuth, B.: Finite volume discretization for poroelastic media with fractures modeled by contact mechanics. Int. J. Numer. Methods Eng. **121**(4), 644–663 (2020). https://doi.org/10.1002/nme.6238
50. Dong, S., Zeng, L., Dowd, P., Xu, C., Cao, H.: A fast method for fracture intersection detection in discrete fracture networks. Comput. Geotech. **98**, 205–216 (2018). https://doi.org/10.1016/j.compgeo.2018.02.005
51. Mallison, B.T., Hui, M.H., Narr, W.: Practical gridding algorithms for discrete fracture modeling workflows. presented at the 12th European Conference on the Mathematics of Oil Recovery, Oxford, UK (2010). https://doi.org/10.3997/2214-4609.20144950
52. Holm, R., Kaufmann, R., Heimsund, B.-O., Øian, E., Espedal, M.S.: Meshing of domains with complex internal geometries. Numer. Linear Algebra Appl. **13**(9), 717–731 (2006). https://doi.org/10.1002/nla.505
53. Berge, R.L., Klemetsdal, Ø.S., Lie, K.-A.: Unstructured Voronoi grids conforming to lower dimensional objects. Comput. Geosci. **23**(1), 169–188 (2019). https://doi.org/10.1007/s10596-018-9790-0
54. Shewchuk, J.R.: Triangle: engineering a 2D quality mesh generator and Delaunay triangulator. In: Applied Computational Geometry: Towards Geometric Engineering, vol. 1148, pp. 203–222 (1996)
55. Si, H.: TetGen, a Delaunay-based quality tetrahedral mesh generator. ACM Trans. Math. Softw. **41**(2), 1–36 (2015). https://doi.org/10.1145/2629697
56. Boffi, D., Brezzi, F., Fortin, M.: Mixed finite element methods and applications. Springer, Berlin (2013)
57. da Veiga, L.B., Brezzi, F., Marini, L.D., Russo, A.: Mixed virtual element methods for general second order elliptic problems on polygonal meshes. ESAIM Math. Model. Numer. Anal. **50**(3), 727–747 (2016). https://doi.org/10.1051/m2an/2015067
58. da Veiga, L.B., Brezzi, F., Marini, L.D., Russo, A.: H(div) and H(curl) -conforming virtual element methods. Numer. Math. **133**(2), 303–332 (2016). https://doi.org/10.1007/s00211-015-0746-1
59. Nordbotten, J.M.: Convergence of a cell-centered finite volume discretization for linear elasticity. SIAM J. Numer. Anal. **53**(6), 2605–2625 (2015). https://doi.org/10.1137/140972792
60. Keilegavlen, E., Nordbotten, J.M.: Finite volume methods for elasticity with weak symmetry. Int. J. Numer. Methods Eng. **112**(8), 939–962 (2017). https://doi.org/10.1002/nme.5538
61. Nordbotten, J.M.: Stable cell-centered finite volume discretization for Biot equations. SIAM J. Numer. Anal. **54**(2), 942–968 (2016). https://doi.org/10.1137/15M1014280
62. Nordbotten, J.M., Keilegavlen, E.: An introduction to multi-point flux (MPFA) and stress (MPSA) finite volume methods for thermo-poroelasticity. arXiv. **2001.01990**, (2020)
63. Hüeber, S., Stadler, G., Wohlmuth, B.I.: A primal-dual active set algorithm for three-dimensional contact problems with coulomb friction. SIAM J. Sci. Comput. **30**(2), 572–596 (2008). https://doi.org/10.1137/060671061
64. Stefansson, I., Berre, I., Keilegavlen, E.: A fully coupled numerical model of thermo-hydro-mechanical processes and fracture contact mechanics in porous media. **arXiv:2008.06289**, (2020)
65. Berge, R.L., Berre, I., Keilegavlen, E., Nordbotten, J.M.: Viscous fingering in fractured porous media. **arXiv:1906.10472**, (2019)
66. Budisa, A., Boon, W., Hu, X.: Mixed-dimensional auxiliary space preconditioners. **arXiv:1910.04704**, (2019)
67. Budiša, A., Hu, X.: Block preconditioners for mixed-dimensional discretization of flow in fractured porous media. Comput. Geosci. (2020). https://doi.org/10.1007/s10596-020-09984-z
68. Ahrens, J., Geveci, B., Law, C.: ParaView: an end-user tool for large data visualization
69. Fumagalli, A., Keilegavlen, E., Scialò, S.: Conforming, non-conforming and non-matching discretization couplings in discrete fracture network simulations. J. Comput. Phys. **376**, 694–712 (2019). https://doi.org/10.1016/j.jcp.2018.09.048
70. Fumagalli, A., Keilegavlen, E.: Dual virtual element methods for discrete fracture matrix models. Oil Gas Sci. Technol. – Rev. D'IFP Energ. Nouv. **74**, 41 (2019). https://doi.org/10.2516/ogst/2019008
71. Stefansson, I., Berre, I., Keilegavlen, E.: Finite-volume discretisations for flow in fractured porous media. Transp. Porous Media. **124**(2), 439–462 (2018). https://doi.org/10.1007/s11242-018-1077-3
72. PorePy implementation with runscripts. https://doi.org/10.5281/zenodo.3374624. (2019)
73. Flemisch, B., Berre, I., Boon, W., Fumagalli, A., Schwenck, N., Scotti, A., Stefansson, I., Tatomir, A.: Benchmarks for single-phase flow in fractured porous media. Adv. Water Resour. **111**, 239–258 (2018). https://doi.org/10.1016/j.advwatres.2017.10.036
74. Mandel, J.: Consolidation des sols (étude mathématique). Geotechnique. **3**(7), 287–299 (1953)
75. Abousleiman, Y., Cheng, A.-D., Cui, L., Detournay, E., Rogiers, J.-C.: Mandel's problem revisited. Geotechnique. **46**(2), 187–195 (1996)
76. Cheng, A.H.-D., Detournay, E.: A direct boundary element method for plane strain poroelasticity. Int. J. Numer. Anal. Methods Geomech. **12**(5), 551–572 (1988). https://doi.org/10.1002/nag.1610120508
77. Mikelić, A., Wang, B., Wheeler, M.F.: Numerical convergence study of iterative coupling for coupled flow and geomechanics. Comput. Geosci. **18**(3–4), 325–341 (2014). https://doi.org/10.1007/s10596-013-9393-8
78. Sneddon, I.N.: Fourier Transforms. Dover Publications, New York (1995)
79. Crouch, S.L., Starfield, A.M.: Boundary Element Methods in Solid Mechanics: with Applications in Rock Mechanics and Geological Engineering. Allen & Unwin, London (1983)