POLITECNICO DI MILANO
School of Industrial and Information Engineering
MSc of Mathematical Engineering

Graduation thesis in Computational Sciences for Engineering

Numerical modeling of fractured porous media subject to compaction

Advisors:

Prof. Luca Formaggia

Dr. Anna Scotti

Andrea Rossetti

798944

Academic Year 2013-2014

Ringrazio la mia relatrice Dottoressa Anna Scotti e il mio correlatore Professor Luca Formaggia per avermi guidato con costanza e disponibilità in questo progetto e il popolo del Tender per avermi ospitato durante questi mesi. In particolare ringrazio Bianca per i suoi preziosi consigli e suggerimenti.
Grazie ai miei genitori, i miei più grandi tifosi, e a mio fratello Nicolò, aspirante ingegnere anche lui al Politecnico, per essere stato sempre al mio fianco.
Inoltre ringrazio i miei amici di Cagliari e i compagni del Poli per aver reso indimenticabili i miei anni scolastici e universitari.
Ci tengo, infine, a ringraziare particolarmente il Professor Sandro Salsa, fondamentale punto di riferimento durante tutto il mio percorso accademico al Politecnico.

Milano, Dicembre 2014 *Andrea*

Abstract

In this work, carried out in collaboration with the computational geoscience group of MOX, starting from previous analysis of the flow through fractured porous rock media, we extend the model taking into account the compaction phenomenon that characterizes the sedimentary layers subject to a progressive burial.
The compaction process, determining changes of the porosity of the rock medium and, in particular, of the fracture aperture, impacts the fluid dynamic behaviour within the sediment, whose pore space is usually filled by water or fluid resulting from the chemical transformation of the organic component of the rock.
Through our analysis we investigate how the presence of an highly permeable fracture and its different physical properties affect the overall flow behaviour, highlighting, moreover, the influence of the choice of the fracture boundary conditions.
The physical problem is characterized by a significant separation of spacial scales, since the aperture of the fracture is considerably smaller compared to size of the domain. Thus, to represent this strong localized heterogeneity we employ a reduced model for the fracture fluid dynamics, avoiding an extremely high computational effort to resolve the scale of the fracture with the grid.
Furthermore to treat the fracture as an immersed interface, thanks to the reduced model, we exploit the Extended Finite Elements (XFEM), that, with a proper enrichment of the element cut by the interface, allow the use of more flexible non-conforming meshes.
We built a suitable solver based on the `C++` finite element library *GetFEM++* to carry out the simulations that are performed in a two dimensional section of a sedimentary layer. We consider the differential problems formulated in an auxiliary fixed domain, derived from the completely compacted configuration of the physical one that, instead, deforms as time elapses. In this way the mesh is built just once at the beginning along with the basis function of the finite element method, reducing the computational costs.

Sommario

Lo studio dei flussi sotterranei è di grande interesse per le le sue applicazioni nelle scienze ambientali. Tuttavia simulazioni realistiche di flussi di Darcy in applicazioni geofisiche sono complicate dalla eterogeneità del mezzo poroso. Il dominio di interesse è, infatti, composto da strati di diversi sedimenti che si sono accumulati durante milioni d'anni e che hanno subito un complesso processo di compattazione e deformazione, determinando una forte variabilità della permeabilità.
Inoltre il complesso stato di sforzo che caratterizza questi sedimenti durante il processo di interramento è spesso causa di fratture nella roccia, e queste regioni fratturate costituiscono forti eterogeneità localizzate, di grande rilevanza per il comportamento fluidodinamico del mezzo poroso. L'effetto delle fratture sul comportamento del fluido è di grande importanza in diversi campi, come lo studio di falde acquifere fratturate, campi geotermici, giacimenti petroliferi e di gas naturale. Le fratture possono essere suddivise in due grandi famiglie: microfratture, e macrofratture, che si estendono per 10 – 100m con spessori di qualche centimetro, o faglie, che si estendono per 100 – 1000m raggiungendo spessori di alcuni metri.
La presenza di microfratture può essere tenuta in conto attraverso tecniche di omogenizzazione, comportando un cambiamento nella permeabilità del sedimento, ma le macrofratture influenzano il flusso, agendo come barriere o vie preferenziali per il fluido, in un modo complesso che non può essere rappresentato correttamente nelle simulazioni numeriche con la semplice omogeneizzazione. Un comportamento simile è associato all'interfaccia tra due differenti strati sedimentari, per esempio una sabbia di grana grossa e uno strato impermeabile di argilla: la superficie di separazione, chiamata orizzonte, può diventare una via preferenziale per il flusso di fluidi, come l'olio ed il gas, che tendono a galleggiare rispetto all'acqua.
La scala spaziale di queste singolarità solitamente necessita di una mesh estremamente fine, comportando costi computazionali proibitivi. Lo spessore tipico delle fratture (dell'ordine di qualche centimetro) e delle faglie (dell'ordine di qualche metro) è quindi molto piccolo rispetto alla lunghezza caratteristica del dominio di interesse, che per queste applicazioni si aggira sulle centinaia di metri per simulazioni relative ai giacimenti, e sul centinaio di chilometri nel caso di simulazioni relative a bacini.
Per affrontare questo problema, una possibilità consiste nell'utilizzare un modello ridotto per rappresentare il flusso all'interno della frattura, rappresentata quindi come un'interfaccia (n - 1) dimensionale immersa nel dominio poroso, con opportune condizioni di accoppiamento tra la frattura e il mezzo. I modelli per la rappresentazione di fratture sono stati approssimati con una grande varietà di metodi, che includono differenze finite, elementi finiti e volumi finiti. In tutti questi casi la frattura e la griglia computazionale del mezzo devono essere coerenti tra di loro, ad esempio la frattura deve costituire un'interfaccia conforme tra due blocchi di mesh. In casi realistici, con fratture numerose e complesse, la conformità della mesh può risultare un vincolo molto rigido. D'Angelo e Scotti in [7] hanno esteso un già noto modello ridotto per il flusso di Darcy in un mezzo fratturato, al caso in cui la mesh della matrice porosa e la mesh della frattura sono indipendenti e non combaciano, arricchendo i classici spazi di elementi finiti di

Raviart-Thomas con delle funzioni di base discontinue sugli elementi tagliati dalla frattura. In questo modo si ottengono gli elementi finiti noti come XFEM (Elementi finiti estesi). In più, Fumagalli e Scotti in [11] hanno valutato l'efficacia e accuratezza del modello ridotto per problemi realistici nel caso multifase, valutando l'errore associato alla riduzione del modello, comparando i risultati con quelli generati dal modello completo.
L'obiettivo di questo progetto è l'estensione dello studio sui mezzi porosi fratturati, considerando il caso di un mezzo fratturato soggetto a compattazione. L'evoluzione del bilancio dello sforzo verticale ha forti implicazioni sulla porosità e, conseguentemente, sulla permeabilità. Per di più il processo di compattazione influenza direttamente l'evoluzione dello spessore della frattura, modificando le sue proprietà e il comportamento fluidodinamico. In questo studio viene analizzato l'impatto di una frattura, caratterizzata da un'elevata permeabilità, sulla fluidodinamica all'interno dello strato sedimentario e si esamina l' influenza delle sue diverse proprietà fisiche. Inoltre, viene messo in evidenza come l'imposizione di diverse condizioni al contorno alle estremità della frattura incida sul risultato delle simulazioni numeriche.
Per effettuare le simulazioni, relative ad una sezione bidimensionale dello strato sedimentario, è stato sviluppato un solutore numerico basato sulla libreria `C++` per gli elementi finiti *GetFEM++*.
Inoltre, uno sviluppo naturale di questo progetto consisterebbe nel tenere conto delle reazioni chimiche che comportano una modifica della porosità. Questi fenomeni, studiati per esempio in [4], includono la precipitazione o dissoluzione minerale e la generazione di petrolio a partire dalla componente organica presente nella roccia, come il kerogen. In una complessa interazione di compattazione meccanica e fluidodinamica, questi processi possono portare ad un'ulteriore riduzione della porosità, come nel caso di deposito minerale, o ad un suo aumento locale quando parte della matrice solida viene convertita in fluido, fenomeno studiato da Giovanardi in [12], dove è stata esaminata la generazione di idrocarburi nella roccia madre, ovvero uno strato di sedimenti ricco di materia organica. In questo caso l'idrocarburo generato viene espulso dalla roccia a causa della compattazione del sedimento dovuta al suo progressivo interramento.
L'accoppiamento di questi processi con un'accurata descrizione del flusso all'interno delle fratture è rilevante per diverse applicazioni: da una parte, permetterebbe di considerare la cementazione di fratture a causa della precipitazione minerale, dall'altra, dal momento che le sorgenti di kerogen si presentano spesso come sottili strisce, il trattamento utilizzato per le fratture precedentemente descritto risulterebbe utile per la rappresentazione di questo tipo di distribuzioni.
Per queste ragioni, anche se l'aggiunta di reazioni chimiche al modello esula dallo scopo di questo lavoro, durante l'analisi è stata utilizzata, dove possibile, una notazione e formulazione generale che possa tenere conto della presenza di roccia reattiva, per facilitare sviluppi successivi del progetto in questa direzione.

Contents

List of Figures

1 Introduction to the problem

The study of underground flows is of great interest for its application to environmental studies. However, realistic simulations of Darcy flow in geophysical applications are quite challenging because of the heterogeneity of the medium. The domain of interest is indeed composed by layers of different sediments that have accumulated over millions of years and which have experienced a complex history of compaction and stress induced deformations, resulting in a strong variability of the permeability.

Moreover the complex stress state experienced during burial history often causes fracturing in the rocks, and fractured regions can be regarded as strongly localized heterogeneities that are very relevant for the flow. The effect of fractures on the flow is important in many different applications such as the study of fractured aquifers, geothermal fields, oil and gas reservoirs and unconventional hydrocarbon sources. Fractures may be broadly divided into two main classes: microfractures and macrofractures, that extends for 10 - 100m with widths of some centimeters, or faults, that extend for $100 - 1000$m, reaching widths of some meters.

The presence of microfractures may be accounted for by homogenization techniques, leading to a change in the effective permeability, but macrofractures influence the flow, acting as barriers or preferential pathways, in a complex way that cannot be reproduced in numerical simulations by simple homogenization. A similar behaviour is associated to the interface between two different sedimentary layers, say a coarse sand layer and an impermeable clay layer: the surface of discontinuity, called horizon, can become a preferential path for the flow of fluids, like oil and gas, that are subject to buoyancy effects in presence of water.
The spatial scale of these features is usually such that a very fine mesh is needed, leading to an extremely high computational cost. The typical thickness of fractures (of the order of centimetres) and faults (of the orders of meters) is indeed very small compared to the characteristic length of the domain of interest that ranges from hundreds of meters for reservoir scale simulations to hundreds of kilometres at basin scale.

One possibility to address this problem is to use a reduced model to account for the flow in fractures, represented as $(n - 1)$ dimensional interfaces immersed in the porous domain, with proper coupling conditions between fracture and medium. Discrete fracture models have been approximated using a variety of numerical methods including finite differences, finite elements and finite volumes. In all the cases above the fracture and the computational grid of the medium have to match, i.e. the fracture is a conforming interface between two mesh blocks.

In realistic cases with numerous and complex fractures, mesh conformity can be a rigid constraint. D'Angelo and Scotti in [7] extended an already known reduced model of Darcy's flow in fractured media to the case where the porous matrix mesh and the fracture mesh are independent and non-matching, enriching the classical Raviart-Thomas finite element basis on the elements cut by the fracture with discontinuous functions. The resulting finite elements are known as XFEM (Extended Finite Elements Method). Moreover Fumagalli and Scotti in [11] assessed the effectiveness and accuracy of the reduced model for realistic problems in the multiphase case, evaluating the error associated

with the model reduction, comparing the results with those provided by the fully resolved model.
The goal of this project is to extend the study of fractured porous media, considering the case of a fractured medium subject to compaction. The evolution of the balance of the vertical stress has strong implications on porosity and, consequently, on permeability. Moreover compaction impacts directly the thickness of the fracture, changing its properties and fluid dynamic behaviour.
Throughout our study we analyse the impact of an highly permeable fracture in the fluid dynamics within the sedimentary layer and investigate the influence of its different physical properties. Moreover, we highlight how the imposition of different boundary conditions at the tips of the fracture affects the outcome of the numerical simulations.
In order to carry out the numerical simulations, that are performed in a two dimensional section of the sedimentary layer, we built a suitable solver based on the `C++` finite element library *GetFEM++*. Furthermore, a natural development of this work would be taking into account chemical reactions that can modify porosity. Those phenomena, studied for instance in [4], include mineral precipitation or dissolution and, for example, oil generation from organic rock component, such as kerogen. In a complex interplay with mechanical compaction and fluid flow, these processes can either further reduce the porosity, as in the case of mineral deposition, or locally increase it when part of the solid matrix is converted into fluid, as analyzed by Giovanardi in [12], where the generation of hydrocarbons in the source rock, i.e. a layer of sediments rich in organic matter, was studied. Here, the compaction of the layer due to the progressive burial causes the expulsion of the generated hydrocarbons from the rock.
The coupling of such processes with an accurate description of the flow in fractures is relevant for different applications: on one hand, it would allow to take into account the cementation of fractures due to mineral precipitations, on the other hand kerogen sources have often the shape of thin layers and the treatment for the fracture, previously described, could be useful to represent this type of spatial distribution. For these reasons, even if the addition of the chemical reactions to the model is out of the scope of this work, in our analysis we will employ, where possible, a general notation and formulation that could account for the presence of reactive rock, to facilitate further developments toward this direction.

2 The model

We consider a two dimensional vertical section of a sedimentary layer and, at this first stage of the analysis, we study a simplified model, neglecting chemical reactions and restricting ourselves to the single phase case, i.e. we consider a pore space fully saturated with water. The region where we set the problem represents a portion of a single sedimentary layer and the space setting of the problem is crucial for the choice of suitable boundary conditions.

2.1 The physical problem

Sediments and sedimentary rocks are porous media, bodies composed of a solid part, called *solid matrix* and a *void space*. The pores are usually connected and a fluid may flow through the void space. The way in which pores are connected and their size determine how permeable a porous medium is for fluid flow, and the volume of the pore space controls its capacity to store fluid.
Moreover, the presence of fractures determines local changes in the permeability of the sedimentary rock, increasing or decreasing it depending on their nature. Therefore, it potentially affects the fluid dynamic behaviour of the porous medium.
The pores are a consequence of the variety of sizes and shapes of the grains the rocks consist of, but it is also the result of a complex interplay of mechanical and possibly chemical processes. We take into account the compaction of the pore space that results from the large overburden to which the rock medium is subject. This process affects the fluid flow by directly changing the permeability of the porous medium and forcing the pore fluid out whenever the pore space is compressed.
The assumptions that are usually made to identify a rock matrix as a porous medium are the following

- the void space of the solid matrix is interconnected;
- the dimensions of the pore space are large compared to the mean free path of fluid molecules. Under this assumption we are allowed to model the fluid in the void space as a continuum;
- the dimensions of the pores are small enough to consider the fluid flow as controlled by adhesive forces at fluid-solid interfaces and cohesive forces at fluid-fluid interfaces.

Under these hypotheses, considering a domain $\Omega \in \mathbb{R}^d$ we define the functions:

$$\chi(\mathbf{x},t) := \begin{cases} 1 \text{ if } & \mathbf{x} \in \text{ void space} \\ 0 \text{ if } & \mathbf{x} \in \text{ solid matrix} \end{cases} \qquad \mathbf{x} \in \Omega$$

and

$$\phi(\mathbf{x},t,r) := \frac{1}{|\mathcal{B}_r(\mathbf{x})|} \int_{\mathcal{B}_r(\mathbf{x})} \chi(\mathbf{y},t) \mathrm{d}\mathbf{y} \qquad \mathbf{x} \in \Omega,$$

where $\mathcal{B}_r(\mathbf{x})$ denotes the d-dimensional ball of center $\mathbf{x}$ and radius r. If there exists r_0 such that $|\frac{\partial \phi}{\partial r}| << 1$ for r in a neighborhood of r_0, then we define *porosity* the field $\phi(\mathbf{x}, t) = \phi(\mathbf{x}, t, r_0)$.
Moreover for the generic phase α, that represent a chemically homogeneous portion of a system under consideration that is separated from other such portions by a definite physical boundary, we define

$$\chi_\alpha(\mathbf{x}, t) := \begin{cases} 1 & \text{if} \quad \mathbf{x} \in \ \alpha \text{ phase} \\ 0 & \qquad \text{otherwise} \end{cases} \qquad \mathbf{x} \in \Omega,$$

and

$$S_\alpha(\mathbf{x}, t, r) := \frac{\int_{\mathcal{B}_r(\mathbf{x})} \chi_\alpha(\mathbf{y}, t) \mathrm{d}\mathbf{y}}{\int_{\mathcal{B}_r(\mathbf{x})} \chi(\mathbf{y}, t) \mathrm{d}\mathbf{y}}$$

We then define *saturation* of phase α the function $S_\alpha(\mathbf{x}, t) := S_\alpha(\mathbf{x}, t, r_0)$, if there exists r_0 such that $|\frac{\partial S_\alpha}{\partial r}| << 1$ for r in a neighborhood of r_0.

2.2 Recasting the problem in a fixed domain

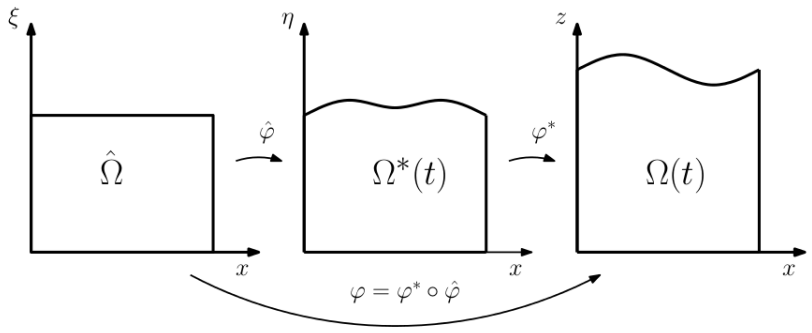

Figure 1: The maps to switch from the different coordinates

Because of the compaction process, the sediment matrix is not fixed and the physical domain changes with time. The equations are then set on a domain changing in time and their numerical solution is thus more complex. A possibility is to resort to a Lagrangian description and write the equations in a fixed domain. We introduce, in order to do so, the domain $\hat{\Omega}$, whose coordinates are (x, ξ), that is obtained from the physical domain $\Omega(t)$ as its completely compacted configuration. Moreover, for the sake of generality if the rock has a reactive part (such as organic matter) that can dissolve in time, we consider $\hat{\Omega}$ as the non-degradable, fully compacted volume. Thus, $\hat{\Omega}$ represents the non-degradable material volume *and is fixed*. We assume that compaction leads only to a vertical movement of the solid matrix. Hence both domains have the same x coordinate. Throughout our analysis we employ a more general notation showing also the general expressions in presence of organic matter, in order to facilitate further developments toward this direction. Thus,

another assumption is that kerogen, or in general the reactive part of the rock, can be considered as dispersed in the solid matrix. Hence, at any point $\mathbf{x}$ inside the domain and at any time t we can define a field $C = C(\mathbf{x}, t)$ that represents the ratio between the volume of kerogen and the initial solid volume, i.e. the solid volume when no chemical reactions have yet occurred. Under these assumptions, the map from the fixed domain to the physical domain is $\boldsymbol{\varphi} : \hat{\Omega} \to \Omega(t)$, $\boldsymbol{\varphi}(x, \xi) = (x, z(\xi, t))$

$$\boldsymbol{\varphi}(x, \xi) = \left(x, \quad z^{top}(x) - \int_{\xi}^{\xi^*} \frac{1 - C_0 + C(\xi')}{(1 - C_0)(1 - \phi(\xi'))} \mathrm{d}\xi' \right), \tag{1}$$

where ξ^* is the height of the domain along the ξ-axis and is computed from the porosity at the initial configuration. We define

$$\mathbf{J} := \nabla \boldsymbol{\varphi} = \begin{bmatrix} 1 & 0 \\ 0 & \partial z / \partial \xi \end{bmatrix},$$

and we set

$$J(x, \xi, t) := \frac{1 - C_0 + C(x, \xi, t)}{(1 - C_0)(1 - \phi(x, \xi, t))}.$$

Since in our simplified case $C(x, \xi, t) = C_0(x, \xi) = 0$, we have

$$J(x, \xi, t) := \frac{1}{(1 - \phi(x, \xi, t))}.$$

This function is the Jacobian determinant of the map and will often appear in the following, as a consequence of the change of coordinates in the integrals, while obtaining the equations.
The advantage of writing the equations and solving the problem on a domain that, differently than the physical one, is fixed, is that we do not need to follow the movement of the domain by deforming the mesh. The mesh can be built once and for all at the beginning of the simulation, as well as the basis functions of the finite element methods.
The following differential equations are written directly on the fixed domain $\hat{\Omega}$ and the boundary conditions will be specified later on.

2.3 The mechanical compaction

In this section we present a mathematical model for the representation of the compaction process, due to the progressive burial of the domain, to which both the bulk medium and fracture are subject.
The porosity ϕ of the rock medium depends on the overload and pore pressure and, in general, on the volume fraction of the reactive rock (i.e. $\phi = \phi(\sigma_e, C)$). By assuming a simple poroelastic behaviour like [18], the porosity is given by

$$\phi(x, \xi, t) = (1 + C(x, \xi, t) - C_0)\phi_0 e^{-\beta \sigma_e(x, \xi, t)} \tag{2}$$

in cases where, $C = C_0 = 0$, it simply becomes the well known Athy's law, derived in [1]

$$\phi(x, \xi, t) = \phi_0 e^{-\beta \sigma_e(x, \xi, t)}.$$

Here β $[Pa^{-1}]$ is the *compressibility* of the rock and $\sigma_e = \sigma_e(\mathbf{x}, t)$ represents the vertical effective stress, which, in the assumption of vertical compaction, can be taken equal to the difference between the bulk pressure σ_T and the fluid pressure p_f,

$$\sigma_e = \sigma_T - p_f.$$

The fluid pressure is defined, in general, as a mean over phase pressures, such as oil pressure p_o and water pressure p_w, weighted with respect to the saturations

$$p_f = S_o p_o + S_w p_w = p_w,$$

where $S_w = S_w(x, \xi, t)$ and $S_o = S_o(x, \xi, t) = 1 - S_w(x, \xi, t)$ are the saturations of water and oil, respectively. Since in our case we consider the pore space initially saturated with water and we do not consider oil generation, we have $S_w = 1$ and $S_o = 0$ for all x, ξ, t.

The bulk pressure at depth z is the pressure due to the weight of sediments and the pore fluids, and assuming that the thickness of the fracture is small enough to ensure the continuity of the bulk pressure, its mechanical contribution can be neglected, yielding

$$\sigma_T(x, \xi, t) = \int_{\xi}^{\xi_{top}} [(1 - \phi)\rho_s + \phi\rho_f] J g \, \mathrm{d}\xi' + \sigma_{top}(t) \tag{3}$$

where σ_{top} is the weight of overlying sedimentary layers and may be variable in time. Note that the density of the fluids ρ_f is a weighted average given, in general, by

$$\rho_f = S_o \rho_o + S_w \rho_w = \rho_w$$

where $\rho_w = \rho_w(x, \xi, t)$ and $\rho_o = \rho_o(x, \xi, t)$ are the densities of water and oil, which may depend on temperature and pressure. Concerning the solid part, ρ_s is given by

$$\rho_s = \frac{(1 - C_0)\rho_r + C\rho_k}{1 - C_0 + C}$$

where ρ_r and ρ_k are the densities of the inert rock and the reactive solid, respectively. In our case $C = C_0 = 0$, thus we have

$$\rho_s = \rho_r$$

The mechanical evolution of the thickness of the fracture l_Γ is taken into account using the law proposed in [16],

$$l_\Gamma = l_{\Gamma,i} + l_{\Gamma,m} \left(e^{-\beta_f \sigma_e^n(x,\xi,t)} - e^{-\beta_f \sigma_{e0}^n(x,\xi)} \right) \tag{4}$$

where $l_{\Gamma,i}$ and $l_{\Gamma,m}$ are respectively the initial and maximum thickness of the fracture, β_f $[Pa^{-1}]$ is the *compressibility* of the fracture, σ_e^n and σ_{e0}^n are respectively the effective stress and the initial effective stress normal to the fracture. Hence the porosity of the fracture ϕ_f is related to its actual and initial thickness and initial porosity ϕ_{f0} as

$$\phi_f = \frac{l_\Gamma}{l_{\Gamma,i}} \phi_{f0}.$$

Since we are not resolving for the full stress tensor, in order to compute the stress component normal to the fracture, that may have different orientations, we estimate the horizontal stress component following [8]. During uniaxial consolidation the horizontal stress $\sigma_{T,H}$ increases as a function of depth but at a reduced rate compared with the vertical stress. Thus we estimate it from the vertical component as

$$\sigma_{T,H} = \frac{\nu}{1-\nu}\sigma_T, \tag{5}$$

where ν is the Poisson coefficient, typically in the range [0.15, 0.30] in the case of rocks. Hence we compute the effective horizontal stress as

$$\sigma_{e,H} = \sigma_{T,H} - p_f.$$

In order to take into account the progressive rock compaction we solve the following equation in $\Omega(t) \times (0, T]$ for the vertical velocity of the solid matrix u_{sz}

$$\frac{\partial}{\partial t}\left((1-\phi)\rho_s\right) + \frac{\partial}{\partial z}\left((1-\phi)\rho_s u_{sz}\right) = Q_s, \tag{6}$$

where Q_s is a source term that may represent the appearance/disappearance of solid, for instance the breakdown of kerogen into oil. In our case $Q_s = 0$. By solving this equation at each time step, we can compute the new position of the nodes of the mesh.
Hence, even if the whole problem is solved in the fixed domain, we can visualize the solutions on the physical domain.

2.4 The fluid dynamic problem in the bulk medium

In this paragraph we present the model that describes the evolution of the fluid pressure and velocity within the bulk medium. We will write the equations directly in the fixed domain, describing with the tilde the reduced variable.
The mass conservation equation for water in the porous medium can be expressed as:

$$\frac{\partial(\rho_w \phi S_w J)}{\partial t} + \nabla \cdot (\rho_w \mathbf{U}) = 0 \qquad \text{in } \hat{\Omega} \times (0, T] \tag{7}$$

We assume no water generation or injection hence there is no source term in the equation. The water flux $\mathbf{U}$ in the rock medium is given by the generalized Darcy law that models the fluid flow through a porous medium in response to a pressure gradient as

$$\mathbf{U} = -J\frac{k_{r,w}\tilde{\mathbf{K}}}{\mu_w}\left(\nabla p_w - \rho_w \mathbf{J}^T \mathbf{g}\right) \qquad \text{in } \hat{\Omega} \times (0, T] \tag{8}$$

with $\mathbf{g} = -g\mathbf{e}_z$. Notice that in the vertical direction we subtract the effect of gravity, hence a fluid pressure gradient equal to the weight of the fluid column determines a null Darcy flux.
Here $k_{r,w}$ is the relative permeability of the water and is a given function of the saturation.

There are several models for the relative permeability in literature. We model it according to the Brooks-Corey relative permeability curve of [5], namely

$$k_{r,w}(S_w) \quad = \quad S_w^3. \tag{9}$$

Notice that if the relative permeability vanishes, then no flow of the corresponding phase occurs. We have denoted with μ_w the viscosity of water which is modeled as constant. We have set $\tilde{\mathbf{K}} := \mathbf{J}^{-1}\mathbf{K}(\phi)\mathbf{J}^{-T}$, where $\mathbf{K}$ is the permeability tensor, that we assume diagonal, and function of the porosity ϕ according to the following relation:

$$\mathbf{K}(\phi) = \mathcal{K}(\phi) \begin{bmatrix} k_{xx} & 0 \\ 0 & k_{zz} \end{bmatrix} \tag{10}$$

where $\mathcal{K}(\phi)$ is chosen following [6]

$$\mathcal{K}(\phi) = \begin{cases} k_0\phi^3 \text{ if} & \phi \geq 0.1 \\ \dfrac{100\ k_0\phi^5}{(1-\phi)^2} \text{ if} & \phi < 0.1 \end{cases}. \tag{11}$$

Note that since the equation is written in the fixed domain, $\tilde{\mathbf{K}}$ is the tensor transformed with $\mathbf{J}$.

2.5 The fluid dynamic problem for the fracture

In this section we show how to derive a suitable reduced problem, able to model the fluid flow inside the fracture, and how to link it with the flow in the bulk medium with proper interface conditions.
Let us introduce the sub-domain Ω_f, which is the part of Ω occupied by the fracture, as represented in figure 2. We assume that the geometry of the fracture fulfils some requirements:

- The domain Ω is divided into three connected subsets, called Ω_1, Ω_2 and Ω_f, such that $\mathring{\Omega}_i \cap \mathring{\Omega}_j = \emptyset \quad for \quad i \neq j \quad and \quad i,j = 1,2,f,$

  $$\bigcup_{i=1,2,f} \bar{\Omega}_i = \bar{\Omega},$$

  and we define $\gamma_i = \partial\Omega_f \cap \partial\Omega_i, \quad i = 1,2$

- we suppose the existence of a non auto-intersecting $n-1$ manifold Γ of class $\mathcal{C}^2$ such that the domain Ω_f can be written as

  $$\Omega_f := \left\{ x \in \Omega : x = \mathbf{s} + r\mathbf{n}_\Gamma, \mathbf{s} \in \Gamma, r \in \left(-\frac{l_\Gamma(\mathbf{s})}{2}, \frac{l_\Gamma(\mathbf{s})}{2} \right) \right\}, \tag{12}$$

  with $l_\Gamma \in \mathcal{C}^2(\Gamma)$, as previously indicated, the thickness of Ω_f and $\mathbf{n}_\Gamma$ the unit normal to Γ. We require that $\exists\ c_1, c_2 \in \mathbb{R}^+$

  $$l_\Gamma(\mathbf{s}) > c_1 \quad and \quad |l'_\Gamma(\mathbf{s})| < c_2$$

with c_2 "small". So basically, we assume that Γ does not differ too much from a straight line if n = 2.

- the domain Ω_f is "thin" compared with Ω_1 and Ω_2, i.e.

$$diam(\Omega_i) \gg \max_{\mathbf{s}\in\Gamma} l_\Gamma(\mathbf{s}) \quad \text{for} \quad i = 1, 2.$$

With this assumption we can identify Ω_f with Γ and we can adopt the approximations $\mathbf{n}_\Gamma \approx \mathbf{n}_1 \approx -\mathbf{n}_2$, $\partial\Gamma \approx \partial\Omega \cap \partial\Omega_f$

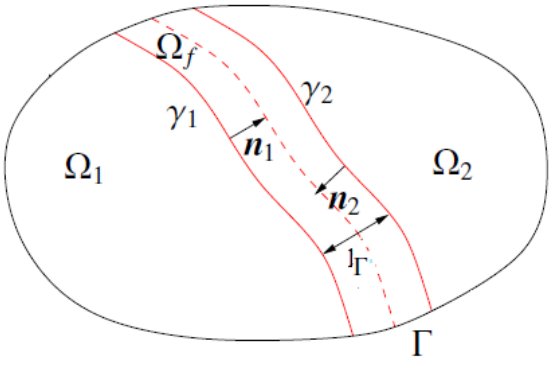

Figure 2: *The domain* Ω*, subdivided in* Ω_1 *and* Ω_2 *by the thin domain* Ω_f *that is assimilated to the curve* Γ*.*

Let us consider the fracture as represented by the line Γ, and denote with $\mathbf{n}_\Gamma$ the normal vector with fixed orientation on Γ from Ω_1 to Ω_2 and with τ_Γ the tangential unit vector on Γ. We model the relation between permeability and porosity in the fracture the same way we did in the case of the rock medium, which determines the following permeability tensor expressed in the local coordinates of the fracture

$$\mathbf{K}_\Gamma(\phi) = \mathcal{K}(\phi) \begin{bmatrix} k_{\Gamma,n} & 0 \\ 0 & k_{\Gamma,\tau} \end{bmatrix} \tag{13}$$

where $\mathcal{K}(\phi)k_{\Gamma,n}$ is the normal permeability and $\mathcal{K}(\phi)k_{\Gamma,\tau}$ is the tangential permeability. Notice that typically the permeability tensor in Γ could be significantly different from that in the rest of the domain Ω, since the porosity of the fracture can be completely different from that of the surrounding rock.
We define the projection matrices along the normal of Γ and on the tangential space as

$$\mathbf{N} := \mathbf{n}_\Gamma \otimes \mathbf{n}_\Gamma \quad \text{and} \quad \mathbf{T} := \tau_\Gamma \otimes \tau_\Gamma = \mathbf{I} - \mathbf{N}, \tag{14}$$

respectively, with the following properties:

$$\mathbf{N} = \mathbf{N}^T, \quad \mathbf{T} = \mathbf{T}^T, \quad \mathbf{NN} = \mathbf{N}, \quad \mathbf{TT} = \mathbf{T}, \quad \text{and} \quad \mathbf{NT} = 0.$$

Moreover it holds

$$\mathbf{N}\mathbf{n}_\Gamma = \mathbf{n}_\Gamma, \quad \mathbf{N}\tau_\Gamma = 0, \quad \mathbf{T}\tau_\Gamma = \tau_\Gamma, \quad \text{and} \quad \mathbf{T}n_\Gamma = 0.$$

Finally, given a generic vector $\mathbf{m}$ we can decompose it along the normal direction and the tangential space as

$$\mathbf{m} = \mathbf{N}\mathbf{m} + \mathbf{T}\mathbf{m} = \mathbf{m}_\mathbf{n} + \mathbf{m}_\tau.$$

Using the projection matrix (14), given a regular scalar function $g: \ \Omega \to \mathbb{R}$ we define the normal and tangential gradient as

$$\nabla_\mathbf{n} g := \mathbf{N}\nabla g \quad \text{and} \quad \nabla_\tau g := \mathbf{T}\nabla g = \nabla g - \nabla_\mathbf{n} g. \tag{15}$$

and given a regular vector function $\mathbf{u}: \ \Omega \to \mathbb{R}^n$ we define the normal and tangential divergence as

$$\nabla_\mathbf{n} \cdot \mathbf{u} := \mathbf{N} : \nabla\mathbf{u} \quad \text{and} \quad \nabla_\tau \cdot \mathbf{u} := \mathbf{T} : \nabla\mathbf{u} = \nabla \cdot \mathbf{u} - \nabla_\mathbf{n} \cdot \mathbf{u}. \tag{16}$$

Let $\mathbf{U}_{f,\tau}$ and p_f be respectively the fluid velocity in the fracture in the tangential direction and the pressure in the fracture. We define the mean flow rate and mean pressure in the fracture as

$$\tilde{\mathbf{U}}(\mathbf{s}) := \int_{-\frac{l_\Gamma(\mathbf{s})}{2}}^{\frac{l_\Gamma(\mathbf{s})}{2}} \mathbf{U}_{f,\tau}(\mathbf{r}) \ \mathrm{d}\mathbf{r}(\mathbf{s}), \tag{17}$$

$$\tilde{p}(\mathbf{s}) := \frac{1}{l_\Gamma(\mathbf{s})} \int_{-\frac{l_\Gamma(\mathbf{s})}{2}}^{\frac{l_\Gamma(\mathbf{s})}{2}} p_f(\mathbf{r}) \ \mathrm{d}\mathbf{r}(\mathbf{s}). \tag{18}$$

2.5.1 The reduced mass conservation equation

Proposition 1. *Under the previous assumptions the reduced mass conservation equation for the fracture in the fixed domain is represented by the following equation*

$$\nabla_\tau \cdot (\rho_w \tilde{\mathbf{U}}) = -l_\Gamma \frac{\partial(\rho_w \phi_f S_w J)}{\partial t} + (\rho_w \mathbf{U}_1 - \rho_w \mathbf{U}_2) \cdot \mathbf{n}_\Gamma \qquad \text{in } \hat{\Gamma} \times (0, T] \tag{19}$$

where $(\mathbf{U}_1 - \mathbf{U}_2) \cdot \mathbf{n}_\Gamma$ *is the jump of the flow normal to the fracture, which may be discontinuous across* Γ.

Proof. In order to derive the reduced problem for the fluid dynamics in the fracture we start from the fluid dynamic problem defined in the full-dimensional fracture Ω_f, we map it in the fixed domain $\hat{\Omega}_f$ and then we reduce it following a procedure similar to the one employed in [9] but with two substantial differences, in our case we have the presence of gravity and the equations are mapped in the fixed domain, thus we have few different

terms which need a proper treatment.
Let us consider the mass conservation equation in the physical domain

$$\nabla \cdot (\rho_w \mathbf{U}) = -\frac{\partial(\rho_w \phi_f S_w)}{\partial t} \quad \text{in } \Omega_f \times (0,T] \tag{20}$$

we map it in the fixed domain according to [12] obtaining

$$\hat{\nabla} \cdot (\rho_w \hat{\mathbf{U}}) = -\frac{\partial(\rho_w \hat{\phi}_f \hat{S}_w J)}{\partial t} \quad \text{in } \hat{\Omega}_f \times (0,T]. \tag{21}$$

From now we will omit the " $\hat{}$ " to simplify the notation. We proceed integrating the mapped mass equation (21) on the thickness of the fracture for every given point $\mathbf{s}^* \in \Gamma$

$$\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \nabla \cdot (\rho_w \mathbf{U}) \, \mathrm{d}\mathbf{r} = -\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \frac{\partial(\rho_w \phi_f S_w J)}{\partial t} \, \mathrm{d}\mathbf{r}.$$

Decomposing the divergence operator along the normal direction and the tangential space we obtain

$$\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \nabla_{\mathbf{n}} \cdot (\rho_w \mathbf{U}) \, \mathrm{d}\mathbf{r} + \int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \nabla_\tau \cdot (\rho_w \mathbf{U}) \, \mathrm{d}\mathbf{r} = -\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \frac{\partial(\rho_w \phi_f S_w J)}{\partial t} \, \mathrm{d}\mathbf{r}.$$

Assuming small changes on the porosity, J, ρ_w and S_w along the fracture thickness, we approximate

$$-\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \frac{\partial(\rho_w \phi_f S_w J)}{\partial t} \, \mathrm{d}\mathbf{r} \approx -l_\Gamma \frac{\partial(\rho_w \phi_f S_w J)}{\partial t},$$

moreover integrating the normal divergence term $\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \nabla_{\mathbf{n}} \cdot (\rho_w \mathbf{U}) \, \mathrm{d}\mathbf{r}$ we obtain

$$\rho_w \mathbf{U}|_{\gamma_2} \cdot \mathbf{n}_\Gamma - \rho_w \mathbf{U}|_{\gamma_1} \cdot \mathbf{n}_\Gamma + \int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \nabla_\tau \cdot (\rho_w \mathbf{U}) \, \mathrm{d}\mathbf{r} = -l_\Gamma \frac{\partial(\rho_w \phi_f S_w J)}{\partial t}.$$

Thanks to previous assumptions on the fracture geometry we have

$$\rho_w \mathbf{U}|_{\gamma_2} \cdot \mathbf{n}_\Gamma \approx -\rho_w \mathbf{U}|_{\gamma_2} \cdot \mathbf{n}_2 = -\rho_w \mathbf{U}_2|_{\gamma_2} \cdot \mathbf{n}_2 \approx \rho_w \mathbf{U}_2|_\Gamma \cdot \mathbf{n}_\Gamma,$$

$$\rho_w \mathbf{U}|_{\gamma_1} \cdot \mathbf{n}_\Gamma \approx \rho_w \mathbf{U}|_{\gamma_1} \cdot \mathbf{n}_1 = \rho_w \mathbf{U}_1|_{\gamma_1} \cdot \mathbf{n}_1 \approx \rho_w \mathbf{U}_1|_\Gamma \cdot \mathbf{n}_\Gamma,$$

so we obtain

$$\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \nabla_\tau \cdot (\rho_w \mathbf{U}) \, \mathrm{d}\mathbf{r} = (\rho_w \mathbf{U}_1 - \rho_w \mathbf{U}_2) \cdot \mathbf{n}_\Gamma - l_\Gamma \frac{\partial(\rho_w \phi_f S_w J)}{\partial t}.$$

Exploiting the properties of tensor $\mathbf{T}$, we have

$$\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \nabla_\tau \cdot (\rho_w \mathbf{U})\ \mathrm{d}\mathbf{r} = \int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \mathbf{T} : \nabla(\rho_w \mathbf{U})\ \mathrm{d}\mathbf{r} = \int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} (\mathbf{T}\mathbf{T}) : \nabla(\rho_w \mathbf{U})\ \mathrm{d}\mathbf{r} =$$

$$= \mathbf{T} : \int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \mathbf{T}\nabla(\rho_w \mathbf{U})\ \mathrm{d}\mathbf{r} \approx \mathbf{T} : \nabla \int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \mathbf{T}(\rho_w \mathbf{U})\ \mathrm{d}\mathbf{r},$$

since we assume ρ_w homogeneous across the thickness of the fracture, and using definitions (16) and (17) we have

$$\nabla_\tau \cdot (\rho_w \tilde{\mathbf{U}}) = (\rho_w \mathbf{U}_1 - \rho_w \mathbf{U}_2) \cdot \mathbf{n}_\Gamma - l_\Gamma \frac{\partial(\rho_w \phi_f S_w J)}{\partial t}.$$

□

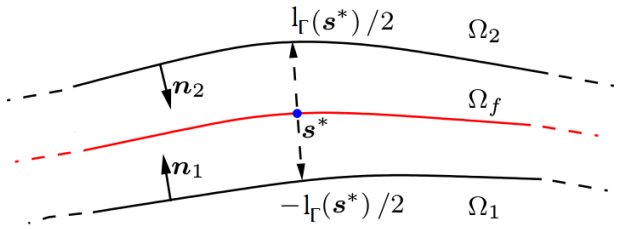

Figure 3: *A representation of the thickness along which we integrate the equation.*

2.5.2 The reduced Darcy equation

Thanks to assumption (13) on the permeability tensor in the fracture, and since $\mathbf{J}$ is diagonal, the following property for the mapped permeability tensor in the fracture $\tilde{\mathbf{K}}_\Gamma := \mathbf{J}^{-1}\mathbf{K}_\Gamma(\phi)\mathbf{J}^{-T}$ holds:
$\tilde{\mathbf{K}}_\Gamma$ can be written as

$$\tilde{\mathbf{K}}_\Gamma = \tilde{K}_{\Gamma,\mathbf{n}}\mathbf{N} + \tilde{K}_{\Gamma,\tau}\mathbf{T}.$$

Moreover we make the following assumptions

- $\tilde{K}_{\Gamma,\mathbf{n}}$, $\tilde{K}_{\Gamma,\tau} \in L^\infty(\hat{\Omega}_f)$ are strictly positive;
- both $\tilde{K}_{\Gamma,\mathbf{n}}$ and $\tilde{K}_{\Gamma,\tau}$ are constant on each cross section of the fracture, i.e. for fixed $\mathbf{s}$.

Consequently, we have

$$\tilde{\mathbf{K}}_\Gamma\mathbf{n} = \tilde{K}_{\Gamma,\mathbf{n}}\mathbf{n} \quad \text{and} \quad \tilde{\mathbf{K}}_\Gamma\tau = \tilde{K}_{\Gamma,\tau}\tau.$$

Proposition 2. *Under the previous assumption the Darcy equation for the fracture in the fixed domain is represented by the following equation*

$$\tilde{\mathbf{U}} = -J\frac{k_{r,w}\tilde{K}_{\Gamma,\tau}}{\mu_w} l_\Gamma \left(\nabla_\tau \tilde{p} - \rho_w \mathbf{T}\mathbf{J}^T \mathbf{g}\right) \qquad in\ \hat{\Gamma} \times (0,T] \tag{22}$$

Proof. Proceeding in the same way of the proof of proposition (1) we start from the Darcy equations in the full-dimensional fracture domain

$$\mathbf{U} = -\frac{k_{r,w}\mathbf{K}_\Gamma}{\mu_w}\left(\nabla p - \rho_w \mathbf{g}\right) \quad \text{in} \quad \Omega_f \times (0,T] \tag{23}$$

requiring, moreover,

$$\begin{cases} \mathbf{U}\cdot\mathbf{n}_j = \mathbf{U}_j \cdot \mathbf{n}_j \\ p = p_j \end{cases} \quad \text{on } \gamma_j. \tag{24}$$

(23) mapped in the fixed domain becomes

$$\hat{\mathbf{U}} = -J\frac{k_{r,w}\tilde{\mathbf{K}}_\Gamma}{\mu_w}\left(\hat{\nabla}\hat{p} - \rho_w \mathbf{J}^T \hat{\mathbf{g}}\right) \quad \text{in} \quad \hat{\Omega}_f \times (0,T]. \tag{25}$$

Applying $\mathbf{T}$ on both sides of the latter equation, and omitting the " $\hat{}$ " we have

$$\mathbf{T}\mathbf{U} = -J\frac{k_{r,w}\tilde{K}_{\Gamma,\tau}}{\mu_w}\mathbf{T}\left(\nabla p - \rho_w \mathbf{J}^T\mathbf{g}\right),$$

that is

$$\mathbf{T}\mathbf{U} = -J\frac{k_{r,w}\tilde{K}_{\Gamma,\tau}}{\mu_w}\left(\nabla_\tau p - \rho_w \mathbf{T}\mathbf{J}^T\mathbf{g}\right),$$

and integrating across the thickness of the fracture for a given point $\mathbf{s}^* \in \Gamma$ we have

$$\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \mathbf{T}\mathbf{U}\ \mathrm{d}\mathbf{r} = \tilde{\mathbf{U}} = -J\frac{k_{r,w}\tilde{K}_{\Gamma,\tau}}{\mu_w}\left(\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \nabla_\tau p - \int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \rho_w \mathbf{T}\mathbf{J}^T\mathbf{g}\right) =$$

$$= -J\frac{k_{r,w}\tilde{K}_{\Gamma,\tau}}{\mu_w}\left(\mathbf{T}\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \nabla p - \mathbf{T}\rho_w \int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \mathbf{J}^T\mathbf{g}\right) \approx$$

$$\approx -J\frac{k_{r,w}\tilde{K}_{\Gamma,\tau}}{\mu_w}\left(\nabla_\tau \int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} p - \mathbf{T}\rho_w \int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \mathbf{J}^T\mathbf{g}\right).$$

Using the definition (18) we finally obtain

$$\tilde{\mathbf{U}} = -J\frac{k_{r,w}\tilde{K}_{\Gamma,\tau}}{\mu_w} l_\Gamma \left(\nabla_\tau \tilde{p} - \rho_w \mathbf{T}\mathbf{J}^T\mathbf{g}\right).$$

□

2.5.3 The interface conditions

Proposition 3. *The problem is closed with the following interface conditions*

$$\eta_n[\mathbf{U}\cdot\mathbf{n}_\Gamma] = \frac{4}{2\xi-1}(\{p\}-\tilde{p}) \qquad on\ \hat{\Gamma} \tag{26}$$

$$\eta_n\{\mathbf{U}\cdot\mathbf{n}_\Gamma\} = [p] + l_\Gamma\{\mathbf{G}\cdot\mathbf{n}_\Gamma\} \qquad on\ \hat{\Gamma} \tag{27}$$

where $\eta_n = \frac{l_\Gamma\mu_w}{Jk_{r,w}\tilde{K}_{\Gamma,\mathbf{n}}}$, $\mathbf{G} = \rho_w\mathbf{J}^T\mathbf{g}$. *Note that we have adopted the following notations for the jump and the average of any function f that may be discontinuous across* Γ

$$[f] := f_1 - f_2, \quad \{f\} := \tfrac{1}{2}(f_1+f_2), \quad \text{where} \quad f_{1,2} = \lim_{\epsilon\to 0^\pm} f(\mathbf{x}-\epsilon\mathbf{n}_\Gamma) \quad \forall\mathbf{x}\in\Gamma.$$

Proof. Let us apply $\mathbf{N}$ on both sides of the equation (25), we obtain

$$\mathbf{NU} = -J\frac{k_{r,w}\tilde{K}_{\Gamma,\mathbf{n}}}{\mu_w}\mathbf{N}\left(\nabla p - \rho_w\mathbf{J}^T\mathbf{g}\right),$$

that is

$$\mathbf{NU} = -J\frac{k_{r,w}\tilde{K}_{\Gamma,\mathbf{n}}}{\mu_w}\left(\nabla_{\mathbf{n}} p - \rho_w\mathbf{N}\mathbf{J}^T\mathbf{g}\right),$$

multiplying it for $\mathbf{n}_\Gamma$ and integrating in the direction normal to the fracture we get

$$\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \mathbf{NU}\cdot\mathbf{n}_\Gamma \,\mathrm{d}\mathbf{r} = -J\frac{k_{r,w}\tilde{K}_{\Gamma,\mathbf{n}}}{\mu_w}\left(\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \nabla_{\mathbf{n}} p\cdot\mathbf{n}_\Gamma - \int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \rho_w\mathbf{N}\mathbf{J}^T\mathbf{g}\cdot\mathbf{n}_\Gamma\right).$$

Integrating the pressure term, and since

$$\mathbf{N}\mathbf{J}^T\mathbf{g}\cdot\mathbf{n}_\Gamma = \mathbf{J}^T\mathbf{g}\cdot\mathbf{N}^T\mathbf{n}_\Gamma = \mathbf{J}^T\mathbf{g}\cdot\mathbf{N}\mathbf{n}_\Gamma = \mathbf{J}^T\mathbf{g}\cdot\mathbf{n}_\Gamma$$

we have

$$\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \mathbf{NU}\cdot\mathbf{n}_\Gamma \,\mathrm{d}\mathbf{r} = -J\frac{k_{r,w}\tilde{K}_{\Gamma,\mathbf{n}}}{\mu_w}\left(p|_{\gamma_2} - p|_{\gamma_1} - \int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \rho_w\mathbf{J}^T\mathbf{g}\cdot\mathbf{n}_\Gamma\right).$$

Thanks to previous assumptions on the fracture geometry we can approximate

$$p|_{\gamma_2} - p|_{\gamma_1} \approx p_2|_\Gamma - p_1|_\Gamma,$$

$$\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \mathbf{NU}\cdot\mathbf{n}_\Gamma \,\mathrm{d}\mathbf{r} \approx \frac{l_\Gamma}{2}\left(\mathbf{NU}|_{\gamma_2}\cdot\mathbf{n}_\Gamma + \mathbf{NU}|_{\gamma_1}\cdot\mathbf{n}_\Gamma\right) = \frac{l_\Gamma}{2}\left(\mathbf{U}_2|_\Gamma\cdot\mathbf{n}_\Gamma + \mathbf{U}_1|_\Gamma\cdot\mathbf{n}_\Gamma\right) = l_\Gamma\{\mathbf{U}\cdot\mathbf{n}_\Gamma\},$$

$$\int_{-\frac{l_\Gamma}{2}}^{\frac{l_\Gamma}{2}} \rho_w\mathbf{J}^T\mathbf{g}\cdot\mathbf{n}_\Gamma \approx l_\Gamma\{\rho_w\mathbf{J}^T\mathbf{g}\cdot\mathbf{n}_\Gamma\}.$$

Setting

$$\eta_n = \frac{l_\Gamma \mu_w}{J k_{r,w} \tilde{K}_{\Gamma,\mathbf{n}}}$$

we finally get

$$\eta_n \{\mathbf{U}\cdot\mathbf{n}_\Gamma\} = [p] + l_\Gamma \{\rho_w \mathbf{J}^T \mathbf{g}\cdot\mathbf{n}_\Gamma\}$$

that is equation (27).
Now let us derive the first interface condition following the procedure of [9]. We approximate the value of the pressure inside the fracture by the following Taylor expansion, regarding the first transversal section in figure 4

$$p(\mathbf{s}^*) = p(\mathbf{x}_1) + \frac{l_\Gamma}{2}\nabla p(\theta_1)\cdot\mathbf{n}_\Gamma, \tag{28}$$

where $\theta_1 = \mathbf{s}^* - \xi_1 \frac{l_\Gamma(\mathbf{s}^*)}{2}\mathbf{n}_\Gamma$ with $\xi_1 \in [0,1]$. The point $\mathbf{x}_1 \in \gamma_1$ is such that $\mathbf{x}_1 = \mathbf{s}^* - \frac{l_\Gamma(\mathbf{s}^*)}{2}\mathbf{n}_\Gamma$. In the second transversal section we approximate the value of the pressure inside the fracture by

$$p(\mathbf{s}^*) = p(\mathbf{x}_2) - \frac{l_\Gamma}{2}\nabla p(\theta_2)\cdot\mathbf{n}_\Gamma, \tag{29}$$

where $\theta_2 = \mathbf{s}^* + \xi_2 \frac{l_\Gamma(\mathbf{s}^*)}{2}\mathbf{n}_\Gamma$ with $\xi_2 \in [0,1]$. The point $\mathbf{x}_2 \in \gamma_2$ is such that $\mathbf{x}_2 = \mathbf{s}^* + \frac{l_\Gamma(\mathbf{s}^*)}{2}\mathbf{n}_\Gamma$. Using relation (24) and assuming a piecewise linear variation of the normal flow inside the fracture $\mathbf{NU} = \mathbf{U_n}$ we have

$$\begin{cases} \mathbf{U_n}(\theta_1)\cdot\mathbf{n}_\Gamma = \xi_1 \mathbf{U}_1\cdot\mathbf{n}_\Gamma + (1-\xi_1)\mathbf{U}_2\cdot\mathbf{n}_\Gamma, \\ \mathbf{U_n}(\theta_2)\cdot\mathbf{n}_\Gamma = \xi_2 \mathbf{U}_2\cdot\mathbf{n}_\Gamma + (1-\xi_2)\mathbf{U}_1\cdot\mathbf{n}_\Gamma. \end{cases} \tag{30}$$

We approximate $p(\mathbf{x}_1) \approx p_1(\mathbf{x}_1)$ and, exploiting the mapped Darcy equation (25) and the properties of the projection matrix, the pressure in the fracture (28) is approximated as

$$p(\mathbf{s}^*) \approx p_1(\mathbf{x}_1) + \frac{l_\Gamma}{2}\nabla p(\theta_1)\cdot\mathbf{n}_\Gamma = p_1(\mathbf{x}_1) + \frac{l_\Gamma}{2}\nabla_{\mathbf{n}} p(\theta_1)\cdot\mathbf{n}_\Gamma =$$

$$= p_1(\mathbf{x}_1) - \frac{\eta_n}{2}\mathbf{U_n}(\theta_1)\cdot\mathbf{n}_\Gamma + \frac{l_\Gamma}{2}(\rho_w \mathbf{J}^T\mathbf{g})(\theta_1)\cdot\mathbf{n}_\Gamma,$$

using (30) and approximating $(\rho_w \mathbf{J}^T\mathbf{g})(\theta_1) = \mathbf{G}(\theta_1) \approx \mathbf{G}_1|_\Gamma$ we get

$$p(\mathbf{s}^*) \approx p_1(\mathbf{x}_1) - \frac{\eta_n}{2}\left(\xi_1\mathbf{U}_1\cdot\mathbf{n}_\Gamma + (1-\xi_1)\mathbf{U}_2\cdot\mathbf{n}_\Gamma\right) + \frac{l_\Gamma}{2}\mathbf{G}_1|_\Gamma\cdot\mathbf{n}_\Gamma =$$

$$= p_1(\mathbf{x}_1) - \frac{\eta_n}{2}\left(\{\mathbf{U}\cdot\mathbf{n}_\Gamma\} + (\xi_1 - \frac{1}{2})[\mathbf{U}\cdot\mathbf{n}_\Gamma]\right) + \frac{l_\Gamma}{2}\mathbf{G}_1|_\Gamma\cdot\mathbf{n}_\Gamma$$

while equation (29) provides

$$p(\mathbf{s}^*) \approx p_2(\mathbf{x}_2) + \frac{\eta_n}{2}\left(\xi_2\mathbf{U}_2\cdot\mathbf{n}_\Gamma + (1-\xi_2)\mathbf{U}_1\cdot\mathbf{n}_\Gamma\right) - \frac{l_\Gamma}{2}\mathbf{G}_2|_\Gamma\cdot\mathbf{n}_\Gamma =$$

$$= p_2(\mathbf{x}_2) + \frac{\eta_n}{2}\left(\{\mathbf{U}\cdot\mathbf{n}_\Gamma\} - (\xi_2 - \frac{1}{2})[\mathbf{U}\cdot\mathbf{n}_\Gamma]\right) - \frac{l_\Gamma}{2}\mathbf{G}_2|_\Gamma\cdot\mathbf{n}_\Gamma.$$

Now, using the interface condition previously derived (27) to substitute $\{\mathbf{U}\cdot\mathbf{n}_\Gamma\}$, we get

$$p(\mathbf{s}^*) \approx \{p\} - \frac{\eta_n(2\xi_1 - 1)}{4}[\mathbf{U}\cdot\mathbf{n}_\Gamma] - \frac{l_\Gamma}{4}[\mathbf{G}\cdot\mathbf{n}_\Gamma]$$

$$p(\mathbf{s}^*) \approx \{p\} - \frac{\eta_n(2\xi_2 - 1)}{4}[\mathbf{U}\cdot\mathbf{n}_\Gamma] - \frac{l_\Gamma}{4}[\mathbf{G}\cdot\mathbf{n}_\Gamma]$$

Since the pressure in the fracture is single valued at $\mathbf{s}^*$, the only possibility is to choose $\xi_1 = \xi_2 = \xi$, then integrating in the transversal section of Ω_f and using the definition (18) for the reduced pressure we obtain the last coupling condition

$$\tilde{p} = \{p\} - \frac{\eta_n(2\xi - 1)}{4}[\mathbf{U}\cdot\mathbf{n}_\Gamma] - \frac{l_\Gamma}{4}[\mathbf{G}\cdot\mathbf{n}_\Gamma].$$

In our case, since we are assuming the aperture of the fracture "small" enough, we consider the gravity jump negligible, obtaining

$$\tilde{p} = \{p\} - \frac{\eta_n(2\xi - 1)}{4}[\mathbf{U}\cdot\mathbf{n}_\Gamma].$$

□

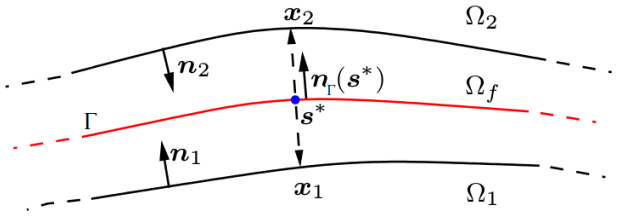

Figure 4: *A representation of the coordinates employed for the Taylor expansion.*

The equations for the flow in the rock medium (7), (8), along with the reduced problem (19), (22) and the interface conditions (26), (27) constitute a system of coupled problems governing the fluid motion in the fractured domain.

$$\begin{cases} \dfrac{\partial(\rho_w\phi S_w J)}{\partial t} + \nabla\cdot(\rho_w\mathbf{U}) = 0 & \text{in } \hat{\Omega}\times(0,T] \\ \mathbf{U} = -J\dfrac{k_{r,w}\tilde{\mathbf{K}}}{\mu_w}\left(\nabla p_w - \rho_w\mathbf{J}^T\mathbf{g}\right) & \text{in } \hat{\Omega}\times(0,T] \\ \nabla_\tau\cdot(\rho_w\tilde{\mathbf{U}}) = -l_\Gamma\dfrac{\partial(\rho_w\phi_f S_w J)}{\partial t} + (\rho_w\mathbf{U}_1 - \rho_w\mathbf{U}_2)\cdot\mathbf{n}_\Gamma & \text{in } \hat{\Gamma}\times(0,T] \\ \tilde{\mathbf{U}} = -J\dfrac{k_{r,w}\tilde{K}_{\Gamma,\tau}}{\mu_w}l_\Gamma\left(\nabla_\tau\tilde{p} - \rho_w\mathbf{T}\mathbf{J}^T\mathbf{g}\right) & \text{in } \hat{\Gamma}\times(0,T] \end{cases} \tag{31}$$

$$\begin{cases} \eta_n[\mathbf{U}\cdot\mathbf{n}_\Gamma] = \dfrac{4}{2\xi-1}(\{p\}-\tilde{p}) & \text{on } \hat{\Gamma} \\ \eta_n\{\mathbf{U}\cdot\mathbf{n}_\Gamma\} = [p] + l_\Gamma\{\mathbf{G}\cdot\mathbf{n}_\Gamma\} & \text{on } \hat{\Gamma} \end{cases} \tag{32}$$

The well posedness of this coupled problem has been proved for $\frac{1}{2} < \xi \leq 1$ in [14].

3 Numerical approximation

In this section we will derive a numerical formulation for the equations in the fixed domains $\hat{\Omega}$ and $\hat{\Gamma}$ illustrated in the previous section and we will suggest a proper splitting strategy. Let us introduce a triangulation $\mathcal{T}_h$ of the domain $\hat{\Omega}$, being h the maximal diameter of the elements of $\mathcal{T}_h$, and a one-dimensional mesh $\tilde{\mathcal{T}}_h$ of the fracture $\hat{\Gamma}$. For the sake of simplicity, we assume that $\hat{\Omega}_h := \bigcup_{K\in\mathcal{T}_h} K = \hat{\Omega}$.
We point out that $\mathcal{T}_h$ may not be conformal with the fracture $\hat{\Gamma}$, so that triangles $K \in \mathcal{T}_h$ may be cut by $\hat{\Gamma}$.

3.1 The equation for the bulk pressure

We start analysing equation

$$\sigma_T(x,\xi,t) = \int_{\xi}^{\xi_{top}} [(1-\phi)\rho_r + \phi\rho_f]Jg \ \mathrm{d}\xi' + \sigma_{top}(t)$$

for the bulk pressure σ_T. It is more convenient to bring this equation to its differential formulation. For this purpose, we derive it with respect to ξ and obtain the following differential problem:

$$\frac{\partial \sigma_T}{\partial \xi} = ((1-\phi)\rho_r + \phi\rho_f)Jg \quad \text{with} \quad \sigma_T(x,\xi_{top},t) = \sigma_{top}(t). \tag{33}$$

Note that the right hand side of the equation depends on the porosity ϕ which changes in time because of the rock compaction. Thus, the problem is indeed time dependent.

Multiplying equation (33) by a test function φ, we have

$$\int_{\hat{\Omega}} \frac{\partial \sigma_T}{\partial \xi}\varphi \ \mathrm{d}\hat{\Omega} = \int_{\hat{\Omega}} f\varphi \ \mathrm{d}\hat{\Omega},$$

where in our case $f = [(1-\phi)\rho_r + \phi\rho_f]Jg$. The assumptions on the coefficients that we will list in the next section grant that the right hand side of the stress differential equation is in $L^\infty(\hat{\Omega}\times(0,T])$.

The discrete formulation is obtained by choosing the discrete space $S_h := \mathbb{P}_1(\hat{\Omega},\mathcal{T}_h) \subset H^1(\hat{\Omega})$.
The numerical solution of this problem requires a stabilization. We propose the strongly consistent *SUPG* stabilization, which consists of the addition of the term

$$\sum_{K\in\mathcal{T}_h} \delta h_K \int_K \left(\frac{\partial \sigma_{Th}}{\partial \xi} - f\right)\frac{\partial \varphi_h}{\partial \xi}\mathrm{d}K$$

to the integral formulation, where h_K is the dimension of the K-th element of the triangulation $\mathcal{T}_h$.

Hence, a stabilized discrete formulation of the problem is:

Discrete formulation:

find $\sigma_{Th}^{n+1} \in S_h$ such that $\sigma_{Th}^{n+1}(x, \xi_{top}) = \sigma_{top}(t^{n+1})$ and

$$\int_{\hat{\Omega}} \frac{\partial \sigma_{Th}^{n+1}}{\partial \xi} \varphi_h \, \mathrm{d}\hat{\Omega} + \sum_{K \in \mathcal{T}_h} \delta h_K \int_K \frac{\partial \sigma_{Th}^{n+1}}{\partial \xi} \frac{\partial \varphi_h}{\partial \xi} \mathrm{d}K = \int_{\hat{\Omega}} f \varphi_h \, \mathrm{d}\hat{\Omega} + \sum_{K \in \mathcal{T}_h} \delta h_K \int_K f \frac{\partial \varphi_h}{\partial \xi} \mathrm{d}K,$$

$\forall \varphi_h \in S_h$.

Following the usual procedure, we can write

$$\sigma_{Th}^{n+1} = \sum_j \sigma_j^{n+1} \varphi_{hj},$$

impose that the equation holds for each element φ_{hi} of the base of S_h, and obtain the algebraic system:

$$\mathbf{A}\sigma^{n+1} = \mathbf{F}$$

where

$$A_{ij} = \int_{\hat{\Omega}} \frac{\partial \varphi_{hj}}{\partial \xi} \varphi_{hi} \, \mathrm{d}\hat{\Omega} + \sum_{K \in \mathcal{T}_h} \delta h_K \int_K \frac{\partial \varphi_{hj}}{\partial \xi} \frac{\partial \varphi_{hi}}{\partial \xi} \mathrm{d}K \tag{34}$$

$$F_i = \int_{\hat{\Omega}} f \varphi_{hi} \, \mathrm{d}\hat{\Omega} + \sum_{K \in \mathcal{T}_h} \delta h_K \int_K f \frac{\partial \varphi_{hi}}{\partial \xi} \mathrm{d}K. \tag{35}$$

The Dirichlet boundary condition at ξ_{top} is imposed directly on the assembled matrix and right hand side vector, by eliminating from A and F the degrees of freedom corresponding to the Dirichlet nodes.

The same discretization technique explained in this paragraph will also be used to solve numerically equation (6) for the mesh nodes. In this case, we will take

$$f = -\frac{\Delta((1-\phi)\rho_r)}{\Delta t}$$

and homogeneous Dirichlet boundary conditions will be imposed at $z = 0$.

3.2 The flow equations

We now analyse the equations governing the fluid flow in the fractured domain managing the fracture problem with the XFEM approach suggested by D'Angelo and Scotti in [7]. Let us consider the one phase fluid dynamic problem in the rock medium and in the fracture completed with the boundary conditions,

$$\begin{cases} \dfrac{\partial(\phi J)}{\partial t} + \nabla \cdot \mathbf{U} = 0 & \text{in} \quad \hat{\Omega} \times (0,T] \\ \mathbf{U} = -J\lambda \tilde{\mathbf{K}} \left(\nabla p - \mathbf{G}\right) & \text{in} \quad \hat{\Omega} \times (0,T] \end{cases} \quad \begin{cases} p = p_D & \text{on } \partial_D \hat{\Omega} \\ \mathbf{U} \cdot \mathbf{n} = h & \text{on } \partial_N \hat{\Omega} \end{cases} \tag{36}$$

$$\begin{cases} \nabla_\tau \cdot \tilde{\mathbf{U}} = -l_\Gamma \dfrac{\partial(\phi_f J)}{\partial t} + (\mathbf{U}_1 - \mathbf{U}_2) \cdot \mathbf{n}_\Gamma & in \ \hat{\Gamma} \times (0,T] \\ \tilde{\mathbf{U}} = -J\lambda \tilde{K}_{\Gamma,\tau} l_\Gamma \left(\nabla_\tau \tilde{p} - \mathbf{TG}\right) & in \ \hat{\Gamma} \times (0,T] \end{cases} \tag{37}$$

$$\begin{cases} \tilde{p} = \tilde{p}_D & \text{on } \partial_D \hat{\Gamma} \\ \tilde{\mathbf{U}} \cdot \tau_\Gamma = 0 & \text{on } \partial_N \hat{\Gamma} \end{cases} \tag{38}$$

The problem is completed with the interface conditions (26), (27).
We assume that the Dirichlet boundary $\partial_D\hat{\Omega}$ and the Neumann boundary $\partial_N\hat{\Omega}$ are such that $\partial_D\hat{\Omega} \cap \partial_N\hat{\Omega} = \emptyset$ and $\partial_D\hat{\Omega} \cup \partial_N\hat{\Omega} = \partial\hat{\Omega}$. $\lambda = \frac{k_{r,w}}{\mu_w}$ is the water mobility. We assumed ρ_w homogeneous and constant in time and we took into account $S_w = 1$.
Let us derive the weak formulation of this problem starting from (36). Let us work formally first and test the equations against two sufficiently smooth test functions q and $\mathbf{v}$. We will specify the functional setting later on.

We have

$$\int_{\hat{\Omega}} \nabla \cdot \mathbf{U}\ q\ \mathrm{d}\hat{\Omega} = \int_{\hat{\Omega}} -\frac{\partial}{\partial t}(\phi J)\ q\ \mathrm{d}\hat{\Omega}$$

$$\int_{\hat{\Omega}} \frac{1}{\lambda J}\tilde{\mathbf{K}}^{-1}\mathbf{U} \cdot \mathbf{v}\ \mathrm{d}\hat{\Omega} = -\int_{\hat{\Omega}} (\nabla p - \mathbf{G}) \cdot \mathbf{v}\ \mathrm{d}\hat{\Omega}$$

Integrating by parts the pressure term of the right hand side of the second equation, we obtain

$$\int_{\hat{\Omega}} \frac{1}{\lambda J}\tilde{\mathbf{K}}^{-1}\mathbf{U} \cdot \mathbf{v}\ \mathrm{d}\hat{\Omega} = \int_{\hat{\Omega}} p\ \nabla \cdot \mathbf{v}\ \mathrm{d}\hat{\Omega} - \int_{\partial\hat{\Omega}} p\ \mathbf{v} \cdot \mathbf{n}\ \mathrm{d}\gamma + \int_{\hat{\Omega}} \mathbf{G} \cdot \mathbf{v}\ \mathrm{d}\hat{\Omega}$$

If we assume that $\mathbf{v} \cdot \mathbf{n} = 0$ on $\partial_N\hat{\Omega}$, we obtain the following equations:

$$\int_{\hat{\Omega}} \nabla \cdot \mathbf{U}\ q\ \mathrm{d}\hat{\Omega} = \int_{\hat{\Omega}} -\frac{\partial}{\partial t}(\phi J)\ q\ \mathrm{d}\hat{\Omega}$$

$$\int_{\hat{\Omega}} \frac{1}{\lambda J}\tilde{\mathbf{K}}^{-1}\mathbf{U} \cdot \mathbf{v}\ \mathrm{d}\hat{\Omega} - \int_{\hat{\Omega}} p\ \nabla \cdot \mathbf{v}\ \mathrm{d}\hat{\Omega} = -\int_{\partial_D\hat{\Omega}} p_D\ \mathbf{v} \cdot \mathbf{n}\ \mathrm{d}\gamma + \int_{\hat{\Omega}} \mathbf{G} \cdot \mathbf{v}\ \mathrm{d}\hat{\Omega}$$

If we approximate the term $\frac{\partial}{\partial t}(\phi J)$ as

$$\frac{\partial}{\partial t}(\phi J) \approx \frac{\Delta(\phi J)}{\Delta t},$$

where we indicate with $\frac{\Delta(\phi J)}{\Delta t}$ a suitable finite difference approximation (such as $\Delta(\phi J) = \phi^* J^* - \phi^{**} J^{**}$ where * and ** mean $n+1$, n, or $n-1$), the following functional setting guarantees that all the previous integrals make sense. Concerning the coefficients and the boundary conditions, we require that:

- $\phi, J, \rho_w, \mu_w \in L^\infty(\hat{\Omega} \times (0,T])$;
- $\tilde{\mathbf{K}} \in (L^\infty(\hat{\Omega} \times (0,T]))^{d\times d}$
  (if $\det(\tilde{\mathbf{K}}) \neq 0$, this condition implies $\tilde{\mathbf{K}}^{-1} \in (L^\infty(\hat{\Omega} \times (0,T]))^{d\times d}$);
- $p_{D,h} \in L^2(0,T; H^{1/2}(\partial_D\hat{\Omega}))$;

Let us introduce the space

$$\mathbf{H}_{div}(\hat{\Omega}) := \{\mathbf{v} \in (L^2(\hat{\Omega}))^d : \nabla \cdot \mathbf{v} \in L^2(\hat{\Omega})\}.$$

After the time discretization, a proper weak formulation for the pressure problem reads:

Weak formulation:

find $\mathbf{U}^{n+1} \in \mathbf{H}_{div}(\hat{\Omega})$ and $p^{n+1} \in L^2(\hat{\Omega})$ such that $\mathbf{U}^{n+1} \cdot \mathbf{n} = h$, and

$$\int_{\hat{\Omega}} \nabla \cdot \mathbf{U}^{n+1} \; q \; \mathrm{d}\hat{\Omega} = \int_{\hat{\Omega}} -\frac{\Delta(\phi J)}{\Delta t} \; q \; \mathrm{d}\hat{\Omega},$$

$$\int_{\hat{\Omega}} \frac{1}{\lambda J} \tilde{\mathbf{K}}^{-1} \mathbf{U}^{n+1} \cdot \mathbf{v} \; \mathrm{d}\hat{\Omega} - \int_{\hat{\Omega}} p^{n+1} \; \nabla \cdot \mathbf{v} \; \mathrm{d}\hat{\Omega} = - \int_{\partial_D \hat{\Omega}} p_D \; \mathbf{v} \cdot \mathbf{n} \; \mathrm{d}\gamma + \int_{\hat{\Omega}} \mathbf{G} \cdot \mathbf{v} \; \mathrm{d}\hat{\Omega},$$

$\forall q \in L^2(\hat{\Omega})$ and $\forall \mathbf{v} \in \{\mathbf{v} \in \mathbf{H}_{div}(\hat{\Omega}) : \mathbf{v} \cdot \mathbf{n} = 0 \; on \; \partial_N \hat{\Omega}\}$.

Notice that, while the Dirichlet boundary condition on pressure is included in the equations, the Neumann boundary condition needs to be embedded in the functional space setting.

A common choice for the finite dimensional spaces for such problem are the lowest order Raviart Thomas elements $\mathbb{RT}_0(\hat{\Omega}, \mathcal{T}_h) \subset \mathbf{H}_{div}(\hat{\Omega})$ for velocity, and the space of the piece-wise constant function $\mathbb{P}_0(\hat{\Omega}, \mathcal{T}_h) \subset L^2(\hat{\Omega})$ for pressure.
Mixed methods have the advantage of approximating the velocity field as a variable of the problem, while in a classic formulation the velocity has to be computed by numerical differentiation of pressure. Moreover, since in a mixed formulation the continuity equation is not integrated by parts, we can also expect it to be satisfied with higher accuracy with respect to classic methods.
However we have allowed the interface Γ to cut the elements and we have employed the extended finite element method (XFEM), based on the technique of enriching the elements cut by an "embedded" interface with discontinuous functions, as exposed in the works by Hansbo on the elasticity problem in domains with fractures [2], [3], [13]. We denote $K_i = K \cap \hat{\Omega}_i \;\; \forall K \in \mathcal{T}_h$ and $\mathcal{G}_h = \{K \in \mathcal{T}_h : K \cap \hat{\Gamma} \neq \emptyset\}$ the collection of elements crossed by the fracture. Let us introduce the finite element spaces that will be used in the set up of the method.
First we define $\forall K \in \mathcal{T}_h \; \mathbb{RT}_0(K_i) = \{\mathbf{U}_h|_{K_i} : \mathbf{U}_h \in \mathbb{RT}_0(K)\}$ as the linear space of the restrictions to K_i of the standard Raviart-Thomas $\mathbb{RT}_0$ local functions. Analogously we define $\mathbb{P}_0(K_i)$. We consider discrete velocities $\mathbf{v}_h$ and pressures q_h in the following spaces,

$$\mathbf{V}_h = \mathbf{V}_{1,h} \times \mathbf{V}_{2,h}, \quad \mathcal{Q}_h = \mathcal{Q}_{1,h} \times \mathcal{Q}_{2,h},$$

$$\mathbf{V}_{i,h} = \{\mathbf{v}_h \in \mathbf{H}_{div}(\hat{\Omega}_i) : \mathbf{v}_h|_{K_i} \in \mathbb{RT}_0(K_i) \; \forall K \in \mathcal{T}_h\}$$
$$\mathcal{Q}_{i,h} = \{q_h \in L^2(\hat{\Omega}_i) : q_h|_{K_i} \in \mathbb{P}_0(K_i) \; \forall K \in \mathcal{T}_h\}.$$

Each discrete velocity $\mathbf{v}_h = (\mathbf{v}_{1,h}, \mathbf{v}_{2,h})$ and pressure $q_h = (q_{1,h}, q_{2,h})$ is thus made of two components, associated to the domains $\hat{\Omega}_i, i = 1, 2$. The discrete variables are discontinuous on $\hat{\Gamma}$, being defined on each part K_i of a cut element $K \in \mathcal{G}_h$ by independent $(\mathbb{RT}_0, \mathbb{P}_0)$ local functions. So, on all cut elements each local function is actually a pair of independent restrictions of the traditional finite element spaces to each of the two subregions. So, basically the finite element basis for the spaces $\mathbf{V}_h$ and $\mathcal{Q}_h$ are obtained from the standard $\mathbb{RT}_0$ and $\mathbb{P}_0$ basis on the mesh replacing each standard basis function living on an element that intersects the interface by its restrictions to $\hat{\Omega}_1$ and $\hat{\Omega}_2$ respectively.

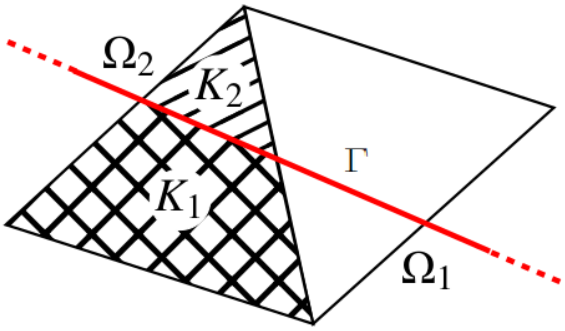

Figure 5: *A representation of the cut elements, divided in two pieces associated to the two domain* Ω_1 *and* Ω_2.

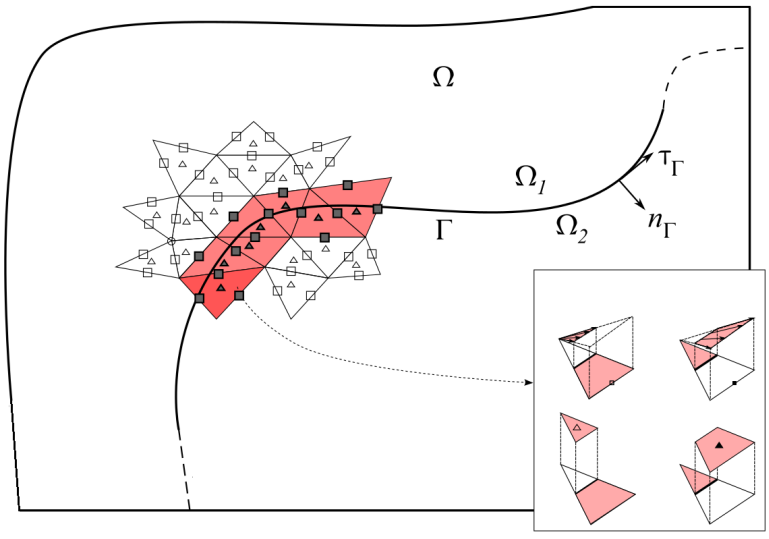

Figure 6: *A representation of the enrichment of the elements cut by the fracture.*

The Neumann boundary condition $\mathbf{U} \cdot \mathbf{n} = h$ on $\partial_N \hat{\Omega}$, which is not included in the

equations yet, is imposed with a Nitsche penalization technique. Thus, we add to the left hand side of the equations the term

$$\int_{\partial_N \hat{\Omega}} \gamma_U (\mathbf{U} \cdot \mathbf{n})(\mathbf{v} \cdot \mathbf{n}) \, \mathrm{d}\gamma,$$

and to the right hand side the term

$$\int_{\partial_N \hat{\Omega}} \gamma_U h(\boldsymbol{\tau} \cdot \mathbf{n}) \, \mathrm{d}\gamma.$$

Where γ_U is a suitable penalization parameter.
Moreover following the same procedure on $\hat{\Omega}_1$, $\hat{\Omega}_2$ separately like in [9] we insert the interface conditions (26), (27) as natural conditions in our variational formulation. The full discretization of the problem in $\hat{\Omega}$ is then

Discrete formulation:

find $\mathbf{U}_h^{n+1} \in \mathbf{V}_h$ and $p_h^{n+1} \in \mathcal{Q}_h$ such that

$$\begin{aligned}
&\int_{\hat{\Omega}} \frac{1}{\lambda_h J_h} \tilde{\mathbf{K}}_h^{-1} \mathbf{U}_h^{n+1} \cdot \mathbf{v}_h \, \mathrm{d}\hat{\Omega} - \int_{\hat{\Omega}} p_h^{n+1} \, \nabla \cdot \mathbf{v}_h \, \mathrm{d}\hat{\Omega} + \int_{\partial_N \hat{\Omega}} \gamma_U (\mathbf{U}_h^{n+1} \cdot \mathbf{n})(\mathbf{v}_h \cdot \mathbf{n}) \, \mathrm{d}\gamma \\
&\quad + \int_{\hat{\Gamma}} \eta_n \{\mathbf{U}_h^{n+1} \cdot \mathbf{n}_\Gamma\} \{\mathbf{v}_h \cdot \mathbf{n}_\Gamma\} \, \mathrm{d}\gamma + \xi_0 \int_{\hat{\Gamma}} \eta_n [\mathbf{U}_h^{n+1} \cdot \mathbf{n}_\Gamma][\mathbf{v}_h \cdot \mathbf{n}_\Gamma] \, \mathrm{d}\gamma \\
&\quad = -\int_{\partial_D \hat{\Omega}} p_D \, \mathbf{v}_h \cdot \mathbf{n} \, \mathrm{d}\gamma + \int_{\hat{\Omega}} \mathbf{G}_h \cdot \mathbf{v}_h \, \mathrm{d}\hat{\Omega} + \int_{\partial_N \hat{\Omega}} \gamma_U h (\mathbf{v}_h \cdot \mathbf{n}) \, \mathrm{d}\gamma \\
&\quad\quad - \int_{\hat{\Gamma}} \eta_n \tilde{p} [\mathbf{v}_h \cdot \mathbf{n}_\Gamma] \, \mathrm{d}\gamma + \int_{\hat{\Gamma}} l_\Gamma \{\mathbf{G}_h \cdot \mathbf{n}_\Gamma\} \{\mathbf{v}_h \cdot \mathbf{n}_\Gamma\} \, \mathrm{d}\gamma,
\end{aligned}$$

$$\int_{\hat{\Omega}} \nabla \cdot \mathbf{U}_h^{n+1} \, q_h \, \mathrm{d}\hat{\Omega} = \int_{\hat{\Omega}} -\frac{\Delta(\phi J)}{\Delta t} \, q_h \, \mathrm{d}\hat{\Omega},$$

$\forall \mathbf{v}_h \in \mathbf{V}_h$ and $\forall q_h \in \mathcal{Q}_h$.

The weak formulation and finite element approximation of the fracture flow problem (37) is obtained following the same method employed for the bulk flow problem (36). In this case we consider standard (not extended) finite element spaces

$$\tilde{\mathbf{V}}_h = \{\tilde{\mathbf{v}}_h \in \mathbf{H}_{div}(\hat{\Gamma}) : \tilde{\mathbf{v}}_h|_{K_i} \in \mathbb{RT}_0(K) \; \forall K \in \tilde{\mathcal{T}}_h\}$$

$$\tilde{\mathcal{Q}}_h = \{\tilde{q}_h \in L^2(\hat{\Gamma}) : \tilde{q}_h|_K \in \mathbb{P}_0(K) \; \forall K \in \tilde{\mathcal{T}}_h\}.$$

The discrete problem in $\hat{\Gamma}$ is thus the following

Discrete formulation:

given the normal velocity jump $[\mathbf{U}_h \cdot \mathbf{n}_\Gamma]$ find $\tilde{\mathbf{U}}_h^{n+1} \in \tilde{\mathbf{V}}_h$ and $\tilde{p}_h^{n+1} \in \tilde{\mathcal{Q}}_h$ such that

$$\int_{\hat{\Gamma}} \frac{1}{\lambda_h J_h l_\Gamma} [\tilde{K}_{\Gamma,\tau}]^{-1} \tilde{\mathbf{U}}_h^{n+1} \cdot \tilde{\mathbf{v}}_h \ \mathrm{d}\gamma - \int_{\hat{\Gamma}} \tilde{p}_h^{n+1} \ \nabla \cdot \tilde{\mathbf{v}}_h \ \mathrm{d}\gamma + \int_{\partial_N \hat{\Gamma}} \gamma_U (\tilde{\mathbf{U}}_h^{n+1} \cdot \tau_\Gamma)(\tilde{\mathbf{v}}_h \cdot \tau_\Gamma) \ \mathrm{d}\gamma$$

$$= - \int_{\partial_D \hat{\Gamma}} \tilde{p}_D \ \tilde{\mathbf{v}}_h \cdot \tau_\Gamma \ \mathrm{d}\gamma + \int_{\hat{\Gamma}} \mathbf{TG}_h \cdot \tilde{\mathbf{v}}_h \ \mathrm{d}\gamma,$$

$$\int_{\hat{\Gamma}} \nabla \cdot \tilde{\mathbf{U}}_h^{n+1} \ \tilde{q}_h \ \mathrm{d}\gamma = \int_{\hat{\Gamma}} -\frac{\Delta(\phi_f J)}{\Delta t} \ \tilde{q}_h \ \mathrm{d}\gamma + \int_{\hat{\Gamma}} [\mathbf{U}_h \cdot \mathbf{n}_\Gamma] \ \tilde{q}_h \ \mathrm{d}\gamma,$$

$\forall \tilde{\mathbf{v}}_h \in \tilde{\mathbf{V}}_h$ and $\forall \tilde{q}_h \in \tilde{\mathcal{Q}}_h$.
Note that discrete problems in the medium and fracture are coupled, since the first one depends on the pressure within the fracture, and the second one on the normal jump of the bulk velocity.

Proceeding with the usual technique, we write $\mathbf{U}_h^{n+1}$, p_h^{n+1}, $\tilde{\mathbf{U}}_h^{n+1}$ and $\tilde{p}_h^{n+1}$ with a proper base of the finite dimensional spaces $\mathbf{V}_h$, $\mathcal{Q}_h$, $\tilde{\mathbf{V}}_h$ and $\tilde{\mathcal{Q}}_h$

$$\mathbf{U}_h^{n+1} = \sum_j U_j^{n+1} \mathbf{v}_{hj} \qquad p_h^{n+1} = \sum_j p_j^{n+1} q_{hj}$$

$$\tilde{\mathbf{U}}_h^{n+1} = \sum_j \tilde{U}_j^{n+1} \tilde{\mathbf{v}}_{hj} \qquad \tilde{p}_h^{n+1} = \sum_j \tilde{p}_j^{n+1} \tilde{q}_{hj}$$

and require that the equations hold for each element of the base of the respective discrete spaces and write for the rock medium

$$\sum_j U_j^{n+1} \left(\int_{\hat{\Omega}} \frac{1}{\lambda_h J_h} \tilde{\mathbf{K}}_h^{-1} \mathbf{v}_{hj} \cdot \mathbf{v}_{hi} \ \mathrm{d}\hat{\Omega} \right) + \sum_j p_j^{n+1} \left(- \int_{\hat{\Omega}} q_{hj} \ \nabla \cdot \mathbf{v}_{hi} \ \mathrm{d}\hat{\Omega} \right)$$

$$+ \sum_j U_j^{n+1} \left(\int_{\partial_N \hat{\Omega}} \gamma_U (\mathbf{v}_{hj} \cdot \mathbf{n})(\mathbf{v}_{hi} \cdot \mathbf{n}) \ \mathrm{d}\gamma \right) + \sum_j U_j^{n+1} \left(\int_{\hat{\Gamma}} \eta_n \{\mathbf{v}_{hj} \cdot \mathbf{n}_\Gamma\} \{\mathbf{v}_{hi} \cdot \mathbf{n}_\Gamma\} \ \mathrm{d}\gamma \right)$$

$$+ \sum_j U_j^{n+1} \left(\xi_0 \int_{\hat{\Gamma}} \eta_n [\mathbf{v}_{hj} \cdot \mathbf{n}_\Gamma][\mathbf{v}_{hi} \cdot \mathbf{n}_\Gamma] \ \mathrm{d}\gamma \right) = \int_{\partial_N \hat{\Omega}} \gamma_U h (\mathbf{v}_{hi} \cdot \mathbf{n}) \ \mathrm{d}\gamma$$

$$- \int_{\partial_D \hat{\Omega}} p_D \ \mathbf{v}_{hi} \cdot \mathbf{n} \ \mathrm{d}\gamma + \int_{\hat{\Omega}} \mathbf{G}_h \cdot \mathbf{v}_{hi} \ \mathrm{d}\hat{\Omega}$$

$$+ \sum_j \tilde{p}_j^{n+1} \left(- \int_{\hat{\Gamma}} \eta_n \tilde{q}_{hj} [\mathbf{v}_{hi} \cdot \mathbf{n}_\Gamma] \ \mathrm{d}\gamma \right) + \int_{\hat{\Gamma}} l_\Gamma \{\mathbf{G}_h \cdot \mathbf{n}_\Gamma\} \{\mathbf{v}_{hi} \cdot \mathbf{n}_\Gamma\} \ \mathrm{d}\gamma,$$

$$\sum_j U_j^{n+1} \left(\int_{\hat{\Omega}} \nabla \cdot \mathbf{v}_{hj} \ q_{hi} \ \mathrm{d}\hat{\Omega} \right) = \int_{\hat{\Omega}} -\frac{\Delta(\phi J)_h}{\Delta t} \ q_{hi} \ \mathrm{d}\hat{\Omega}$$

and for the fracture

$$\sum_j \tilde{U}_j^{n+1} \left(\int_{\hat{\Gamma}} \frac{1}{\lambda_h J_h l_\Gamma} [\tilde{K}_{\Gamma,\tau}]^{-1} \tilde{\mathbf{v}}_{hj} \cdot \tilde{\mathbf{v}}_{hi} \ \mathrm{d}\gamma \right) + \sum_j \tilde{p}_j^{n+1} \left(- \int_{\hat{\Gamma}} \tilde{q}_{hj} \ \nabla \cdot \tilde{\mathbf{v}}_{hi} \ \mathrm{d}\gamma \right)$$

$$+ \sum_j \tilde{U}_j^{n+1} \left(\int_{\partial_N \hat{\Gamma}} \gamma_U (\tilde{\mathbf{v}}_{hj} \cdot \tau_\Gamma)(\tilde{\mathbf{v}}_{hi} \cdot \tau_\Gamma) \ \mathrm{d}\gamma \right) = - \int_{\partial_D \hat{\Gamma}} \tilde{p}_D \ \tilde{\mathbf{v}}_{hi} \cdot \tau_\Gamma \ \mathrm{d}\gamma + \int_{\hat{\Gamma}} \mathbf{T}\mathbf{G}_h \cdot \tilde{\mathbf{v}}_{hi} \ \mathrm{d}\gamma,$$

$$\sum_j \tilde{U}_j^{n+1} \left(\int_{\hat{\Gamma}} \nabla \cdot \tilde{\mathbf{v}}_{hj} \ \tilde{q}_{hi} \ \mathrm{d}\gamma \right) = \int_{\hat{\Gamma}} - \frac{\Delta(\phi_f J)}{\Delta t} \ \tilde{q}_{hi} \ \mathrm{d}\gamma + \sum_j U_j^{n+1} \left(\int_{\hat{\Gamma}} [\mathbf{v}_{hj} \cdot \mathbf{n}_\Gamma] \ \tilde{q}_{hi} \ \mathrm{d}\gamma \right).$$

The equations above can be solved together, written in the following algebraic form

$$\begin{bmatrix} A & B^T & \mathbf{0} & E \\ -B & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \tilde{A} & \tilde{B}^T \\ -E^T & \mathbf{0} & -\tilde{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{U}^{n+1} \\ \mathbf{p}^{n+1} \\ \tilde{\mathbf{U}}^{n+1} \\ \tilde{\mathbf{p}}^{n+1} \end{bmatrix} = \begin{bmatrix} I \\ F \\ \tilde{I} \\ \tilde{F} \end{bmatrix}. \tag{39}$$

Here

$$\begin{aligned}
A_{ij} &= \int_{\hat{\Omega}} \frac{1}{\lambda_h J_h} \tilde{\mathbf{K}}_h^{-1} \mathbf{v}_{hj} \cdot \mathbf{v}_{hi} \ \mathrm{d}\hat{\Omega} + \int_{\partial_N \hat{\Omega}} \gamma_U (\mathbf{v}_{hj} \cdot \mathbf{n})(\mathbf{v}_{hi} \cdot \mathbf{n}) \ \mathrm{d}\gamma \\
&+ \int_{\hat{\Gamma}} \eta_n \{\mathbf{v}_{hj} \cdot \mathbf{n}_\Gamma\} \{\mathbf{v}_{hi} \cdot \mathbf{n}_\Gamma\} \ \mathrm{d}\gamma + \xi_0 \int_{\hat{\Gamma}} \eta_n [\mathbf{v}_{hj} \cdot \mathbf{n}_\Gamma][\mathbf{v}_{hi} \cdot \mathbf{n}_\Gamma] \ \mathrm{d}\gamma \\
B_{ij} &= \int_{\hat{\Omega}} -\nabla \cdot \mathbf{v}_{hj} \ q_{hi} \ \mathrm{d}\hat{\Omega} \\
\tilde{A}_{ij} &= \int_{\hat{\Gamma}} \frac{1}{\lambda_h J_h l_\Gamma} [\tilde{K}_{\Gamma,\tau}]^{-1} \tilde{\mathbf{v}}_{hj} \cdot \tilde{\mathbf{v}}_{hi} \ \mathrm{d}\gamma + \int_{\partial_N \hat{\Gamma}} \gamma_U (\tilde{\mathbf{v}}_{hj} \cdot \tau_\Gamma)(\tilde{\mathbf{v}}_{hi} \cdot \tau_\Gamma) \ \mathrm{d}\gamma \\
\tilde{B}_{ij} &= \int_{\hat{\Gamma}} -\nabla \cdot \tilde{\mathbf{v}}_{hj} \ \tilde{q}_{hi} \ \mathrm{d}\gamma \\
E_{ij} &= \int_{\hat{\Gamma}} \eta_n \tilde{q}_{hj} [\mathbf{v}_{hi} \cdot \mathbf{n}_\Gamma] \ \mathrm{d}\gamma \\
I_i &= \int_{\partial_N \hat{\Omega}} \gamma_U h (\mathbf{v}_{hi} \cdot \mathbf{n}) \ \mathrm{d}\gamma - \int_{\partial_D \hat{\Omega}} p_D \ \mathbf{v}_{hi} \cdot \mathbf{n} \ \mathrm{d}\gamma \\
&+ \int_{\hat{\Omega}} \mathbf{G}_h \cdot \mathbf{v}_{hi} \ \mathrm{d}\hat{\Omega} + \int_{\hat{\Gamma}} l_\Gamma \{\mathbf{G}_h \cdot \mathbf{n}_\Gamma\} \{\mathbf{v}_{hi} \cdot \mathbf{n}_\Gamma\} \ \mathrm{d}\gamma \\
F_i &= \int_{\hat{\Omega}} - \frac{\Delta(\phi J)_h}{\Delta t} \ q_{hi} \ \mathrm{d}\hat{\Omega} \\
\tilde{I}_i &= \int_{\partial_D \hat{\Gamma}} -\tilde{p}_D \ \tilde{\mathbf{v}}_{hi} \cdot \tau_\Gamma \ \mathrm{d}\gamma + \int_{\hat{\Gamma}} \mathbf{T}\mathbf{G}_h \cdot \tilde{\mathbf{v}}_{hi} \ \mathrm{d}\gamma, \\
\tilde{F}_i &= \int_{\hat{\Gamma}} - \frac{\Delta(\phi_f J)}{\Delta t} \ \tilde{q}_{hi} \ \mathrm{d}\gamma.
\end{aligned}$$

Notice that the blocks E and E^T are due to the interface conditions that couple the two problems. Since the basis functions defining E_{ij} are related to different meshes (although the integral is computed on $\hat{\Gamma}$, hence using the fracture mesh), in general an interpolation has to be performed between the bulk mesh $\mathcal{T}_h$ covering $\hat{\Omega}$ and the fracture mesh $\tilde{\mathcal{T}}_h$ on $\hat{\Gamma}$. We solve the system (39) directly using the LU factorization with the package "SuperLU" provided by the Gmm++ library.

3.3 The splitting strategy

The full problem of flow and compaction is therefore coupled, non linear and a fully implicit approach would be too onerous. In this paragraph we describe the splitting strategy that we use in order to solve the whole problem. Let us define the linear operator Σ that associates to ϕ_h the numerical solution σ_{Th}^{n+1} to the stress problem whose coefficients are evaluated in ϕ_h:

$$\sigma_{Th}^{n+1} = \Sigma(\phi_h)$$

We introduce the linear operator $\mathcal{P}$ that associates to the functions, ϕ_h, ϕ_{fh}, l_Γ, and $\Delta(\phi J)_h$ the solutions $\mathbf{U}_h^{n+1}$, p_h^{n+1}, $\tilde{\mathbf{U}}_h^{n+1}$ and $\tilde{p}_h^{n+1}$ of the fluid dynamic problem, where the coefficients are evaluated in ϕ_h, ϕ_{fh}, l_Γ:

$$(\mathbf{U}_h^{n+1}, p_h^{n+1}, \tilde{\mathbf{U}}_h^{n+1}, \tilde{p}_h^{n+1}) = \mathcal{P}\left(\phi_h, \phi_{fh}, l_\Gamma, \Delta(\phi J)_h\right).$$

The initial conditions that we need are the initial porosity of the non-compacted state ϕ_0, the initial porosity of the fracture ϕ_f^0, the initial pressure p^0, the initial aperture of the fracture l_Γ^0, the initial bulk pressure σ_T^0, and initial effective stress σ_e^0. Known, p^0, l_Γ^0 we compute ϕ^0, σ_T^0 and σ_e^0 through some fixed point iterations, so that the initial configuration is an equilibrium one.

Then we can start the time iterations (each time step consist in an inner loop of fixed point iterations). We start by computing the bulk pressure

$$\sigma_T^{n+1} = \Sigma(\phi^n).$$

We can then compute the effective stress as $\sigma_e^{n+1} = \sigma_T^{n+1} - p^n$, the porosities and the fracture aperture as

$$\phi^{n+1} = \phi_0 e^{-\beta\sigma_e^{n+1}},$$

$$l_\Gamma^{n+1} = l_\Gamma^0 + l_{\Gamma,m}\left(e^{-\beta_f\sigma_{e,N}^{n+1}} - e^{-\beta_f\sigma_{e,N}^0}\right),$$

$$\phi_f^{n+1} = \frac{l_\Gamma^{n+1}}{l_\Gamma^0}\phi_f^0.$$

Finally we compute the velocities $\mathbf{U}^{n+1}$, $\tilde{\mathbf{U}}^{n+1}$ and pressures p^{n+1}, $\tilde{p}^{n+1}$:

$$(\mathbf{U}^{n+1}, p^{n+1}, \tilde{\mathbf{U}}^{n+1}, \tilde{p}^{n+1}) = \mathcal{P}\left(\phi^{n+1}, \phi_f^{n+1}, l_\Gamma^{n+1}, \Delta(\phi J)\right),$$

where we choose $\Delta(\phi J) = \phi^{n+1} J(C^0, C^0, \phi^{n+1}) - \phi^n J(C^0, C^0, \phi^n)$, and iterate until convergence is achieved for each time step. The stopping criterion is based on the variations of the Jacobian $J(C, C^0, \phi)$, which accounts for the deformation of the physical domain. Thus, set $\mathbf{J}_k^{n+1}$ the vector of the DOFs of the finite element function $J(C^0, C^0, \phi_k^{n+1})$ we check for each fixed point iteration $k < K_{max}$ if

$$\frac{||\mathbf{J}_k^{n+1} - \mathbf{J}_{k-1}^{n+1}||_2}{||\mathbf{J}_k^{n+1}||_2} < tol.$$

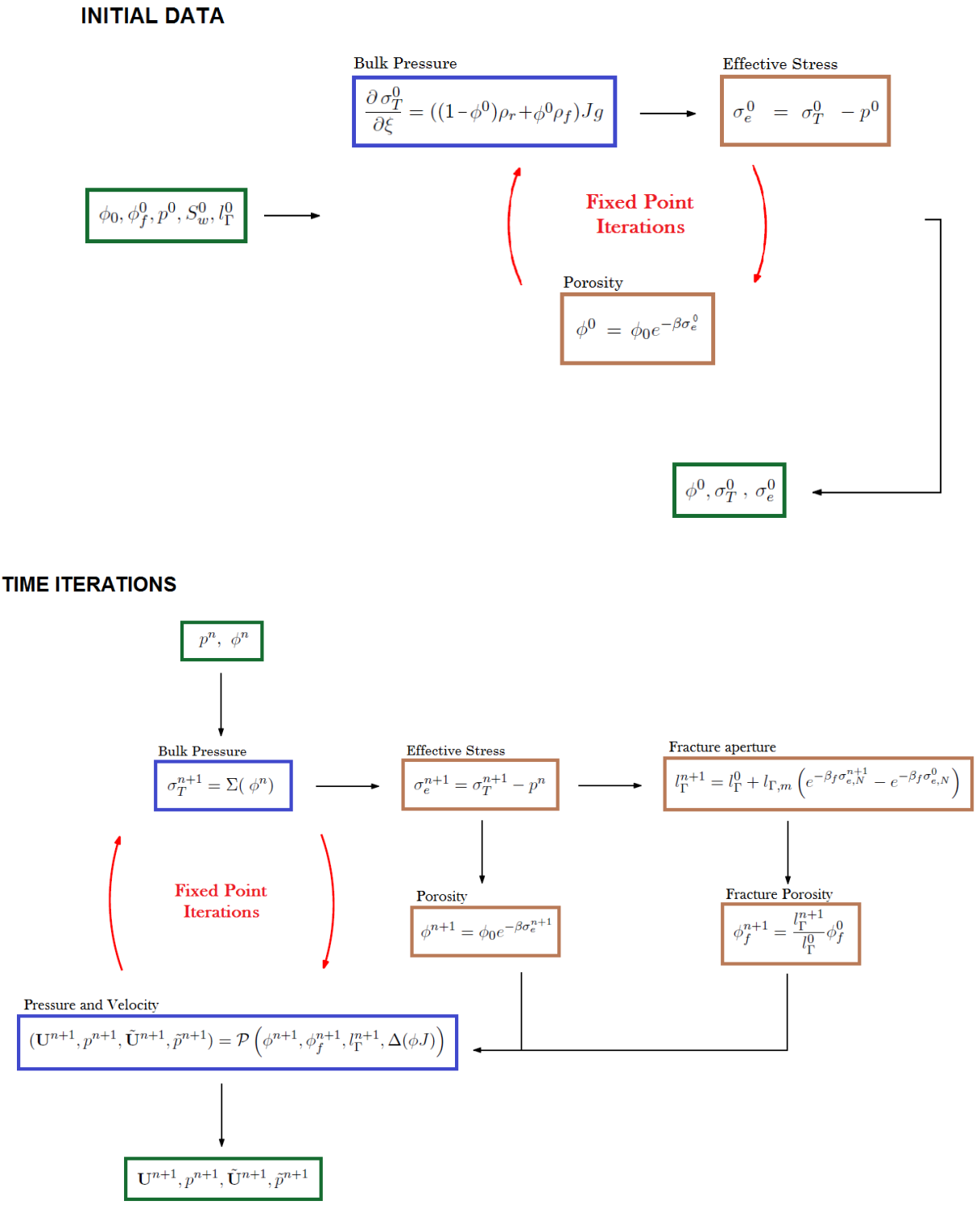

Figure 7: *A scheme of the splitting strategy employed.*

4 Implementation

4.1 Overview

In this section we will describe the structure of the code we implemented to solve the problem analysed in the previous sections.
The starting points of the code were two pre-existing programs. The first one, developed by Bianca Giovanardi in her master thesis work [12], is able to simulate the coupled problem of fluid flow and mechanical compaction in a sedimentary layer, the second one, developed by Alessio Fumagalli and Anna Scotti, related to their works [10], [11], is able to solve the purely fluid dynamic problem in a domain cut by fractures. Thus, both programs have their specific capabilities, useful to solve our target problem, and as well their own limitations. To solve the coupled problem of our interest we will merge the two programs, leveraging their different strengths and building a clean and structured framework handy for the users.

The code is based on *GetFEM++*, a `C++` finite element library. The library includes the tools for the import of meshes and for the construction of regular ones, as well as the usual tools for finite element such as assembly procedures for PDEs and interpolation methods. *GetFEM++* includes *Gmm++*, that is a generic `C++` template library for sparse, dense and skyline matrices.

First of all let us go through some *GetFEM++* terminology. The *mesh* is an object composed of *convexes*. Convexes can be simple line segments, prisms, tetrahedrons, curved triangles, and so on. They all have an associated *reference convex*. E.g. for segments, this will be the $[0, 1]$ segment, for triangles this will be the canonical triangle $(0, 0) - (0, 1) - (1, 0)$. All convexes of the mesh are constructed from the reference convex through a *geometric transformation*. In order to define the geometric transformation, the geometrical nodes are defined on the reference convex. The geometric transformation maps these nodes into the mesh nodes. On the mesh, a set a basis functions is defined: the finite element (*FEM*). The basis functions are attached to some geometrical points, where the *degrees of freedom* are located. The set of all basis functions on the mesh forms the basis of a vector space, on which the PDE will be solved. Obviously, the FEM have to be defined for both the unknown functions and the data. The finite element methods involve the computation of integrals of these basis functions on the convexes (and faces of convexes), approximated using appropriate quadrature formulas. Hence, to each convex is attached an *integration method* along with the FEM. The process of construction of a global linear system from integrals of basis functions on each convex is the *assembly* procedure. A mesh with a set of FEM attached to its convexes is called a *mesh_fem* object in *GetFEM++*, and a mesh with a set of integration methods attached to its convexes is called a *mesh_im* object.

Since the two previous programs have been developed for two different applications they are also characterized by a completely different structure. Four new classes were built in order to merge the the two programs, extend their capabilities and blend them in a common interface.
The two files `functions.cpp` and `functions.h` contain the constants and the constitutive relations, as well as some other useful functions. A data file `data` allows to set the technical

data, such as the finite elements funtions, the mesh description, the penalization and stabilization parameters, and the most important physical information related to the rock medium and fracture, such as the depth, the permeability parameter and the porosity of the non-compacted initial configuration.

4.2 The classes

The four new objects have different task but all of them exploit smart pointers to efficiently link together the different classes of the two previous programs.
We have the *MeshHandlerX*, an object that manages the construction and compaction of the meshes were the differential problems are solved.
Then we have the *IC* that takes care of the setting of the initial conditions.
Finally we have the *StressHandler* and *Darcy*, two objects that, through a similar public interface, that includes the methods `create`, `init`, `assembly` and `solve`, set and solve respectively the differential problem related to the stress and the flow. They also manage other tasks related to the problems, such as the updates of the coefficients and sources as well as the export of the solutions in the in the Visualization Toolkit format.
Both, *StressHandler* and *Darcy*, contain two smart pointers to the *MeshHandlerX* and *IC*, initialised by their constructors, to have a direct link to the mesh and the initial conditions data. The method `create` initializes the smart pointer that manages the object used to solve a specific block of the problem. `init` initialises the setting of the block including the finite element methods for the solution and the coefficients and the integration methods used to compute the integrals during the assembling. The method `assembly` assemblies the algebraic system associated to the differential problem, using the assembling tools provided by *GetFEM++*. Each brick of the matrix is built calling the associated operator (defined in the correspondent `operators` file), then `assembly` organizes the bricks to compose the full algebraic system. The method `solve` solves the algebraic system with the super LU technique provided by *Gmm++*.
The different classes read the needed input data through a *GetPot* object which is one of the input of the constructors.
We would like to point out that the updates of the shared pointers are performed with `make_shared` which is considerably faster because it can use a single allocation for both the object and its corresponding control block, eliminating a significant portion of the shared_ptr construction overhead reducing one of the major efficiency complaints about shared_ptr.
Notice that showing the public interfaces of the objects in the next pages we omitted the "get" methods, used to have access to its private members, for brevity.

4.2.1 Class `MeshHandlerX`

The `MeshHandler` (previously developed by Fumagalli-Scotti) was constructed mainly to build or import the mesh, set the FEM for velocity, pressure, coefficients and, given a pointer to a `FractureSet`, to cut the mesh. This is its public interface

```
public:
    MeshHandler ( const GetPot& dataFile,
                  const std::string& sectionDomain = "" );

    void setUpMesh ( );

    void setUpRegions ( const FracturesSetPtr_Type& fracture );

    void setUpFEM ( );

    void computeMeshMeasures ( );

    void
    printCuttedElements ( const std::string& vtkFolder = "vtk/",
                          const std::string& fileName = "CuttedElements" ) const;
```

The object `MeshHandlerX` extends the capabilities of the `MeshHandler`, as we can see from its public interface

```
public:

    //! @name Constructor
    //@{

    //! \brief The constructor.
    /**
     *  @param dataFile             GetPot object for the input reading.
     *  @param sectionDomain        The input section of the medium data.
     *  @param dataTime             The input section of the time data.
     */

    MeshHandlerX ( const GetPot& dataFile,
                   const std::string& sectionDomain = "", const std::string&
                        dataTime = "" );
        //@}

        //! @name Core methods
    //@{
    //! \brief Sets the mesh.
    void setUpMesh ( );

        //! \brief Sets the regions of the domain.
    void setUpRegions ( const FracturesSetPtr_Type& fracture );

        //! \brief Sets the FEM.
    void setUpFEM ( );

        //! \brief Compute h^-1 used to impose the boundary condition with Nitsche
            penalisation.
    void computeMeshMeasures ( );

        //! \brief Saves the cutted elements in a .vtk file.
    void
    printCuttedElements ( const std::string& vtkFolder = "vtk/",
                          const std::string& fileName = "CuttedElements" ) const;

    //! \brief Computes the positions of the nodes of the fixed (compacted) mesh.
    void computeFixPos ( const scalarVector_Type & C_0, const scalarVector_Type &
        phi_old_curr, std::string folder );

    //! \brief Compacts the mesh changing the position of the nodes.
```

```
    void compactMesh ( const scalarVector_Type & fixedPos, const getfem::mesh_fem &
         mf_s );

    //! \brief Performs the progressive compaction of the mesh.
    void moveMesh(const scalarVector_Type & C_0, const scalarVector_Type & phi_old,
         const scalarVector_Type & phi_new, const scalarVector_Type & z_old);

    //! \brief Compute the compacted mesh with Triangle.
    void triangle ( );

    //! \brief Compute the positions of the nodes of the physical mesh from the
        compacted mesh.
    void computePhysPos ( const getfem::mesh_fem & mf_s_old, const
        scalarVector_Type & C_0, const scalarVector_Type & phi_old_curr, std::
        string folder );

    //! \brief Un-compacts the mesh changing the position of the nodes.
    void uncompactMesh ( const getfem::mesh& fixmesh, const scalarVector_Type &
        physPos, const getfem::mesh_fem & mf_s );

    //! \brief Compute the height of the compacted domain.
    void computeH ( );
    //@}
```

First of all the `MeshHandlerX` is able to build meshes of arbitrary dimensions from the mesh input data that it reads. Moreover it can manage the progressive compaction of the mesh, step by step with the `moveMesh` method, and the complete compaction with the two methods `computeFixPos` (computes the positions of the nodes of the fixed [compacted] mesh) and `compactMesh` (updates the nodes given the fixed positions). The complete compaction process was split in two methods to grant more freedom to the user. For example our application involves the use of two `MeshHandlerX`, one associated to the fixed [compacted] mesh (denoted simply as "mesh") where we solve the differential problems, and another one associated to the moving mesh (denoted as "Z_mesh") that compacts gradually. So the computation of the fixed position is performed at the beginning on "Z_mesh" which has the FEM already set, and then its fixed position output is used as input to call `compactMesh` on "mesh".
Both `moveMesh` and `computeFixPos` have to solve a problem like (6) for the rock velocity. To this aim `MeshHandlerX` contains a smart pointer to a `problemZpde` object (part of the previous implementation), which solves a partial differential equation of type

$$\begin{cases} \dfrac{\partial \sigma}{\partial z} = f & \text{in } \Omega \\ \sigma = \tilde{\sigma} & \text{on } \Gamma_D. \end{cases}$$

Moreover, differently from the previous implementation, it is able also to set and manage the FEM for the stress.

```
private:
    // shared_ptr<MeshHandler_Type>
    MeshHandlerPtr_Type M_MeshHP;

    // integration method for stress
    std::string M_integrationTypeStress;
    getfem::mesh_im M_integrationMethodStress;
```

```
    // mesh_fem for stress
    std::string M_fEMTypeStress;
    getfem::mesh_fem M_meshFEMStress;

    // The SUPG parameter
    scalar_type M_delta;

    // The timestep
    scalar_type M_delta_t;

    // The position of the compacted mesh
    scalarVector_Type M_fixedPos;

    scalarVector_Type M_z_new;

    // The data regarding the Stress
    getfem::pfem M_pFETypeStress;

    getfem::pintegration_method M_pintegrationTypeStress;

    scalar_type M_K0;

    // shared_ptr<problemZpde>
    problemZpdePtr_Type M_problemZpde;

    // Vector containing the coordinates of the nodes of the border
    std::vector<std::pair<scalar_type, scalar_type> > M_nodes;

    // Vector containing the ids of the points of the border segments
    std::vector<std::pair< int, int > > M_segments;

    //prompt command for Triangle
    std::string M_triangle;

    //flags for Triangle
    std::string M_triangleFlags;

    //file for Triangle execution
    std::string M_triangleFile;

    //Height of the compacted domain
    scalar_type M_H;
```

4.2.2 Class `IC`

The class `IC` has the task to set and store the initial conditions as well as to read the physical input data, allowing us to keep the main clean and compact.

```
public:

    //! @name Constructor
    //@{

    //! \brief The constructor.
    /**
     *  @param meshHXP              Smart pointer to the Mesh Handler Extended.
     *  @param dataFile             GetPot object for the input reading.
     *  @param sectionDomain        The input section of the medium data.
     *  @param dataFracDomain       The input section of the fracture data.
```

```
 */

IC ( const MeshHandlerXPtr_Type & meshHXP, const GetPot& dataFile, const std::
    string& sectionDomain = "", const std::string& dataFracDomain = "");

//@}

//! @name Core methods
//@{
//! \brief Sets the initial conditions for the variable that do not need fixed
    point iterations.
void setInitialConditions ( );
//! \brief Overload of the method that is able to use an arbitrary initial
    distribution of kerogen passed as a functor.
    template<class Type>
void setInitialConditions (Type const & C )
{
    // Initial temperature, depth and stress boundary condition
    T = 20. + depth_0*gradientT;
    sigma_bc = 2500.*g*depth_0;


    // Water saturation
    gmm::resize(S_w, mf_coef.nb_dof()); gmm::clear(S_w);
    for (size_type i = 0; i < mf_coef.nb_dof(); i++)
            S_w[i] = 1.;

    // Oil saturation
    gmm::resize(S_o, mf_coef.nb_dof()); gmm::clear(S_o);
    for (size_type i = 0; i < mf_coef.nb_dof(); i++)
        S_o[i] = 0.;

    // Kerogen concentration
    gmm::resize(C_0, mf_s.nb_dof()); gmm::clear(C_0);
    for (size_type i = 0; i < mf_s.nb_dof(); i++)
        C_0[i] = C(mf_s.point_of_basic_dof(i)[0], mf_s.point_of_basic_dof(i)
            [1]);

    // Pressure
    gmm::resize(p_0_s, mf_s.nb_dof()); gmm::clear(p_0_s);
    for (size_type i = 0; i < mf_s.nb_dof(); i++) {
        scalar_type z = mf_s.point_of_basic_dof(i)[1];
        p_0_s[i] = water_density*g*(depth_0 + Z_top - z);
    }
    gmm::resize(p_0, mf_p.nb_dof()); gmm::clear(p_0);
    getfem::interpolation(mf_s, mf_p, p_0_s, p_0);

    // Porosity
    gmm::resize(phi_old_prev, mf_s.nb_dof()); gmm::clear(phi_old_prev);
    for (size_type i = 0; i < mf_s.nb_dof(); i++)
        phi_old_prev[i] = phi_0;

    // Stress
    gmm::resize(sigma_eff_0, mf_s.nb_dof()); gmm::clear(sigma_eff_0);

    gmm::resize(phi_old_curr, mf_s.nb_dof()); gmm::clear(phi_old_curr);

}

//! \brief Sets the initial conditions for porosity and stress with fixed point
     iterations.
void setInitialPhiAndSigma ( );
```

```
    //! \brief Checks the convergence of the fixed point iterations.
    inline bool checkConvergence ( ) const
    {
        return sqrt(norm_diff/norm) < tol_fixed_point;
    }

    //! \brief Updates the porosity.
    inline void updatePhi ( )
    {
        phi_old_prev = phi_old_curr;
    }

    //! \brief Updates the boundary conditions for the stress
    inline void updateSigmaBC ( )
    {
        sigma_bc = sigma_bc + 2500.*g*(M_meshHXP->getDeltaT())*sedimentVelocity;
    }

    //! \brief Sets saturation and initial porosity for the fracture
    void setInitialFracVal ( const FracturesSetPtr_Type & fractures );

    //! \brief Updates the changed dofs after the new mesh is built with Triangle
    void updateFEM ( );
    //@}
```

The method `setInitialConditions` sets the initial conditions for the bulk medium pressure p (and eventually for saturations S_w, S_o, and kerogen concentration C) that do not need fixed point iteration in order to be initialized. In order keep the code more general for further applications, the overload of this method was performed to support arbitrary initial concentration C, that is taken in input by the method as a functor.
On the other hand the method `setInitialPhiAndSigma` sets the initial conditions for the bulk medium porosity ϕ, bulk pressure σ and effective stress σ_e that need some fixed point iterations, keeping the initial pressure constant with the hydrostatic value, to start from a configuration of equilibrium. The stress problem is solved by a `problemZpde` object that, again, is controlled by the `IC` from a smart pointer stored as an attribute of the class.
The `IC` also takes care of computing the target norms and checking the convergence achievement inside the fixed point loop. Moreover `setInitialFracVal` sets the initial data regarding the fracture, like the saturation and porosity. Finally the `IC` has two methods to update the porosity, used for the fixed point iterations, and the boundary conditions used by the the `StressHandler` , which change with the progressive burial of the domain.

```
private:

    //Mesh Handler Extended
    MeshHandlerXPtr_Type M_meshHXP;

    //FEM for the coefficients
    getfem::mesh_fem mf_coef;

    //FEM for the stress
    getfem::mesh_fem mf_s;

    //FEM for the pressure
    getfem::mesh_fem mf_p;

    //Water Saturation
```

```
scalarVector_Type S_w;
scalarVector_Type S_w_frac;

    //Oil Saturation
scalarVector_Type S_o;

//Kerogen concentration
scalarVector_Type C_0;

//Pressure
scalarVector_Type p_0_s;
scalarVector_Type p_0;

//Porosity
scalarVector_Type phi_old_prev;
scalarVector_Type phi_old_curr;

//Stress
scalarVector_Type sigma_eff_0;
scalarVector_Type sigma_0;

//temperature
scalar_type T;
scalar_type gradientT;

//Initial depth
scalar_type depth_0;

//stress boundary condition
scalar_type sigma_bc;

scalar_type Z_top;

//The SUPG parameter
scalar_type delta;

//Norms for the fixed point iterations
scalar_type norm_diff;
scalar_type norm;

//Tollerance fot the fixed point iterations
scalar_type tol_fixed_point;

//Compressibility for the rock medium
scalar_type beta;

//Initial porosity
scalar_type phi_0;

//Initial porosity for the fracture
scalar_type phiF_0;

//Initial Porosity for etaN
scalarVector_Type phi_fracN0;

//Initial Porosity for etaT
scalarVector_Type phi_fracT0;

scalar_type sedimentVelocity;

//Initial thickness
scalar_type M_bi;
```

```
    //Max thickness
    scalar_type M_bm;

    //Fracture compressibility
    scalar_type M_betaf;

    problemZpdePtr_Type M_problemZpde;

    //First component of the versor tangent to the fracture
    scalar_type M_tau1;

    //second component of the versor tangent to the fracture
    scalar_type M_tau2;

    //First component of the versor normal to the fracture
    scalar_type M_n1;

    //second component of the versor normal to the fracture
    scalar_type M_n2;

    //Poisson coef
    scalar_type M_nu;
```

4.2.3 Class `StressHandler`

The `StressHandler` object has the task to solve the equation of the type (33) for the stress, and again this is accomplished through a smart pointer pointing to a `problemZpde` object. It contains, stored within its attributes, a smart pointer to the `MeshHandlerX` and one to the `IC` that give it direct access to the initial conditions, the input data and the mesh where we want to solve the differential problem. Its constructor also takes as input a `FractureSet` object which is needed in order to update some quantities related to the fracture, like its thickness.

```
public:

    //! @name Constructor
    //@{

    //! \brief The constructor.
    /**
     *  @param meshHXP          Smart pointer to the Mesh Handler Extended.
     *  @param ICP              Smart pointer to the Initial Conditions Handler.
     *  @param fractures        Smart pointer to the fracture set.
     */
     StressHandler ( const MeshHandlerXPtr_Type& meshHXP,
                     const ICPtr_Type& ICP,
                     const FracturesSetPtr_Type& fractures);
    //@}

    //! @name Core methods
    //@{

    //! \brief Set the initial stress used to compute the fracture thickness.
    void computeInitialData ( );

    //! \brief Compute rhs for the Stress problem.
    void computeData (const scalarVector_Type& phi_new_prev );
```

```
    //! \brief Creates the ProblemZpde object.
    void createStressProblem ( );




    //! \brief Sets the type of the mesh (w.r.t. the getfem classification of
        geometric transformations) and the mesh itself, the integration method, the
         finite element space, the finite element space for coefficients. All the
        input coefficients must be finite element functions of the type set.

    inline void init ( )
    {

        M_problemZpde->setMesh(M_meshHXP->getMeshType(), M_meshHXP->getMesh());
        M_problemZpde->setIntegrationMethod(M_meshHXP->getIntegrationMethodStress()
            );
        M_problemZpde->setFiniteElement(mf_s);
        M_problemZpde->setFiniteElementCoefficients(mf_coef);
    }

    //! \brief Assemblies the algebraic system associated to the weak formulation
        of the problem.
    inline void assembly ( )
    {
        M_problemZpde->assembly();
    }

    //! \brief Solves the algebraic system built with the function assembly.
    inline void solve ()
    {
        M_problemZpde->solve();
    }

    //! \brief Exports the matrix associated to the algebraic system built with the
         function assembly.
    inline void exportMatrix()
    {
        M_problemZpde->exportMatrix();
    }

    //! \brief Exports solution of the problem in a .vtk file.
    inline void exportVtk(std::string folder)
    {
        M_problemZpde->exportVtk(folder);
    }

    //! \brief Computes the effective stress.
    void computeStrEff(const scalarVector_Type& P_new_prev);

    //! \brief Updates the thickness of the fracture.
    void computeAperture( );

    //@}
```

The method `computeInitialData` computes, from the stress initial condition provided by the `IC`, the initial stress values in the dof of the fracture needed in order to update the fracture thickness at each time step.
The method `computeData` sets the right hand side of the differential problem.

As previously anticipated the core methods related to the differential problem are `createStressProblem`, that updates the pointer to `problemZpde` through the use of make_shared, `init`, `assembly` and `solve` which were already illustrated.
Moreover the `StressHandler` is responsible, given the pressure computed by `Darcy`, for updating the effective stress.
It also takes care of the mechanical compaction of the fracture, computing its current thickness with `computeAperture`.

We complete its description with the list of its private attributes

```
private:

    //Mesh Handler Extended
    MeshHandlerXPtr_Type M_meshHXP;

    // IC
    ICPtr_Type M_ICP;

    //Fracture Set
    FracturesSetPtr_Type M_fractures;

    //Auxiliary initial effective stress
    scalarVector_Type M_sigma_eff_old_coefN;

    //Auxiliary initial effective stress
    scalarVector_Type M_sigma_eff_old_T;

    //Matrix to switch from mf_coef to the pressure FE of the fracture
    sparseMatrixPtr_Type M_fractureMediumInterpolationMatrix;

    scalarVector_Type M_rho_f_s;

    scalarVector_Type M_rhs_coef;

    problemZpdePtr_Type M_problemZpde;

    scalarVector_Type M_sigma_new_curr;

    scalarVector_Type M_P_new_prev_s;

    scalarVector_Type M_sigma_eff_new_curr;

    //The fracture thickness used to compute etaT
    scalarVector_Type M_b_etaT;

    //The fracture thickness used to compute etaN
    scalarVector_Type M_b_etaN;

    getfem::mesh_fem mf_coef;

    getfem::mesh_fem mf_s;

    getfem::mesh_fem mf_p;
```

4.2.4 Class `Darcy`

The class `Darcy` extends a pre-existing class that was able to build and solve the system (39) associated to the full fluid dynamic problem.

```
public:

    public:

    DarcyFractured ( const MediumDataPtr_Type& medium,
                     const MeshHandlerPtr_Type& mesh,
                     const BCHandlerPtr_Type& bcHandler,
                     const FracturesSetPtr_Type& fractures,
                     const ExporterPtr_Type& exporter );
    void init ( );

    void assembly ( );

    void solve ( );
```

`Darcy` is able to implement realistic permeabilities, porosities and source terms and also to take into account the gravity effect, that was neglected in the previously implemented class `DarcyFractured`

```
public:

    //! @name Constructor
    //@{

    //! \brief The constructor.
    /**
     *  @param meshHXP          Smart pointer to the Mesh Handler Extended.
     *  @param ICP              Smart pointer to the Initial Conditions Handler.
     *  @param bcHandler        Smart pointer to the Boundary Conditions Handler.
     *  @param fractures        Smart pointer to the fracture set.
     *  @param exporter         Smart pointer to the Exporter.
     */

    Darcy ( const MediumDataPtr_Type& medium,
                     const MeshHandlerXPtr_Type& meshHXP,
                     const ICPtr_Type& ICP,
                     const BCHandlerPtr_Type& bcHandler,
                     const FracturesSetPtr_Type& fractures,
                     const ExporterPtr_Type& exporter );
    //@}

    //! @name Core methods
    //@{
    //! \brief Sets finite element, permeability, integration methods and select
        the boundaries.
    inline void init ( )
    {
        M_darcyFractured->init();
    }

    //! \brief Assemblies the algebraic system associated to the weak formulation
        of the problem.
    inline void assembly ( )
    {
        M_darcyFractured->assembly();
    }

    //! \brief Solves the algebraic system built with the function assembly.
    inline void solve ()
    {
        M_darcyFractured->solve();
    }
```

```
    //! \brief Exports the pressure solution of the problem in a .vtk file.
    inline void exportVtk (const std::string& str)
    {
        M_darcyFractured->exportVtk(str);

    }

    //! \brief Creates the Darcy Fractured object.
    void createDarcyFrac( const scalar_type& depth );

    //! \brief Coumputes rh, gravity, permeability.
    void computeData (const scalarVector_Type& phi_new_curr, const
        scalarVector_Type& phi_old, const scalarVector_Type& b_etaN, const
        scalarVector_Type& b_etaT, const sparseMatrixPtr_Type &
        fractureMediumInterpolationMatrix );
    //@}
```

The core interface includes the methods `createDarcyFrac`, which updates `DarcyFracturedPtr` through the use of make_shared, `init`, `assembly` and `exportVtk`. Moreover, given the thickness of the fracture from the `StressHandler`, it manages the computation of the right hand side, gravity terms and the permeabilities of the medium and fracture.

```
private:

    MeshHandlerXPtr_Type M_meshHXP;

    DarcyFracturedPtr_Type M_darcyFractured;

    ICPtr_Type M_ICP;

    MediumDataPtr_Type M_medium;

    BCHandlerPtr_Type M_bcHandler;

    FracturesSetPtr_Type M_fractures;

    ExporterPtr_Type M_exporter;

    scalarVector_Type M_KK_xx;

    scalarVector_Type M_KK_xz;

    scalarVector_Type M_KK_zz;

    scalarVector_Type M_etaN;

    scalarVector_Type M_etaT;

    scalar_type M_depth;

    scalarVector_Type M_G_c;

    scalarVector_Type M_Gf_c;

    size_type M_etaTsize;

    scalarVector_Type M_rh_coef;

    scalarVector_Type M_rh_F;
```

```
    scalarVector_Type M_phi_fracTold;

    getfem::mesh_fem mf_coef;

    getfem::mesh_fem mf_s;

    getfem::mesh_fem mf_p;
```

4.3 Functions and data

All the given functions and constants are specified in the files `functions` which was kept and extended from the previous implementation. These data are kept separated, in order to simplify the procedure of changing constitutive relation, such as the porosity law or the relative permeabilities. We report the file `function.h`. Most of the functions are implemented twice: once for the scalar variables and once for the vectorial ones.

```
//-----------------------------------------------------------------------------
// Some given coefficients
    -----------------------------------------------------------------------------
//-----------------------------------------------------------------------------

// The densities and viscosities
const scalar_type water_viscosity = 0.001;  // [Pa s]
const scalar_type oil_viscosity = 0.002;    // [Pa s]
const scalar_type water_density = 1000.;    // [kg/m3]
const scalar_type oil_density = 750.;       // [kg/m3]
const scalar_type kerogen_density = 1150.;  // [kg/m3]
const scalar_type rock_density = 2500.;     // [kg/m3]

// The coefficients for the arrhenius law
const scalar_type arrheniusFactor = 1E12;           // [1/s]
const scalar_type activationEnergy = 200.0*1000.0;  // [J/mol]
const scalar_type gasConstant = 8.314472;           // [J/ (K mol)]

// The gravity acceleration
const scalar_type g = 9.81;                         // [m/s2]

// The coefficient for the permeability function
const scalar_type DarcyTOm2 = 0.987*1E-12;      // conversion factor from Darcy to
    squared meters

// The permeability tensor
const scalar_type k_xx = 1.;                    // [-]
const scalar_type k_xz = 0.;                    // [-]
const scalar_type k_zz = 1.;                    // [-]

// The coefficients for the capillary pressure function
const scalar_type PD = 1E6;                     // [Pa]
const scalar_type acentric_factor = 0.664;      // [-]
const scalar_type m = 0.48508 + 1.55171*acentric_factor - 0.151613*acentric_factor*
    acentric_factor; // [-]

//-----------------------------------------------------------------------------
// Some given functions
    -----------------------------------------------------------------------------
//-----------------------------------------------------------------------------

// Arrehnius constant for kerogen breakdown as a function of temperature [C]
```

```
inline scalar_type arrheniusK (scalar_type T){
    return arrheniusFactor*exp(-activationEnergy/(gasConstant*(T+273.0)));
}

// The capillary pressure function [Pa] and its first derivative wrt Sw [Pa]
scalar_type capillary_pressure (const scalar_type &Sw);
scalar_type capillary_pressure_prime (const scalar_type &Sw);

// Functions that compute water mobility [(Pa s)^-1]
std::vector<scalar_type> lambda_w (const std::vector<scalar_type> &S_w);

inline scalar_type lambda_w (const scalar_type &S_w){
    return S_w*S_w*S_w/water_viscosity;
}

// Functions that compute the first derivative of water mobility [(Pa s)^-1]
std::vector<scalar_type> lambda_prime_w (const std::vector<scalar_type> &S_w);

inline scalar_type lambda_prime_w (const scalar_type &S_w){
    return 3.*S_w*S_w/water_viscosity;
}

// Functions that compute oil mobility [(Pa s)^-1]
std::vector<scalar_type> lambda_o (const std::vector<scalar_type> &S_o);

inline scalar_type lambda_o (const scalar_type &S_o){
    return S_o*S_o*(1. - (1. - S_o)*(1. - S_o))/oil_viscosity;
}

// Functions that compute the first derivative of oil mobility [(Pa s)^-1]
std::vector<scalar_type> lambda_prime_o (const std::vector<scalar_type> &S_o);

inline scalar_type lambda_prime_o (const scalar_type &S_o){
    return 2.*S_o*(1. - (1. - S_o)*(1. - S_o))/oil_viscosity + S_o*S_o*(2.*(1. -
        S_o))/oil_viscosity;
}

// Functions that compute the porosity at t^n+1 [-]
std::vector<scalar_type> computePorosity (const std::vector<scalar_type> &C_0,
                                          const scalar_type &phi_0,
                                          const std::vector<scalar_type> &C_new,
                                          const std::vector<scalar_type>&sigma_new,
                                          const scalar_type beta,
                                          const std::vector<scalar_type>&integral);

std::vector<scalar_type> computeAthyPorosity (const std::vector<scalar_type> &C_0,
                                              const scalar_type &phi_0,
                                              const std::vector<scalar_type>
                                                    &C_new,
                                              const std::vector<scalar_type>
                                                    &sigma_new,
                                              const scalar_type beta);

// Function that computes the density of the solid matrix [kg/m3]
inline scalar_type solidMatrixDensity (scalar_type C, scalar_type C_0) {
    return ((1. - C_0)*rock_density + C*kerogen_density)/(1. - C_0 + C);
}

// Function that computes the fluid density [kg/m3]
std::vector<scalar_type> fluidDensity (std::vector<scalar_type> S_o);

// Function that computes the concentration at t^n+1 [m3_ker/m3_rock]
```

```
std::vector<scalar_type> computeConcentration (const std::vector<scalar_type> &
    C_old, const std::vector<scalar_type> K_new, const scalar_type delta_t);

// Functions that compute the Jacobian determinant of the trasformation from the
    phyisical domain to the reference one [-]
inline scalar_type J (const scalar_type &C, const scalar_type &C_0, const
    scalar_type &phi) {
    return (1. - C_0 + C)/((1. - C_0)*(1. - phi));
}

std::vector<scalar_type> J (const std::vector<scalar_type> &C, const std::vector<
    scalar_type> &C_0, const std::vector<scalar_type> &phi);

// Functions that compute the scalar permeability (i.e. the function that
    multiplies the permeability tensor) [m2]
scalar_type scalarPermeability (const scalar_type &phi, const scalar_type &K0);

std::vector<scalar_type> scalarPermeability (const std::vector<scalar_type> &phi,
    const scalar_type &K0);

// Function that computes the numerical flux [kg/(m2 s)]
scalar_type flux_function(const scalar_type &So, const scalar_type &un, const
    scalar_type &gn, const scalar_type &phi, const scalar_type &K0);

// Function that computes the first derivative of the numerical flux wrt So [kg/(m2
     s)]
scalar_type flux_function_prime(const scalar_type &So, const scalar_type &un, const
     scalar_type &gn, const scalar_type &phi, const scalar_type &K0);

//-------------------------------------------------------------------------------
// Some useful functions
    -------------------------------------------------------------------------------
//-------------------------------------------------------------------------------

// A function that computes the index of the node of the mesh associated to the i-
    th dof of the fem mf.
size_type dof_to_node_index (const size_type &i, const getfem::mesh &mesh, const
    getfem::mesh_fem &mf);

// A function that converts an integer number into a string containing that number
std::string convertInt(size_type number);

// Read a chart containing the values of function pi_w for 1000 values of S_w
// equidistributed in [0, 1]
void read_table(std::vector<scalar_type> &pi_w_table, std::vector<scalar_type> &
    S_w_table);

// A function to compute pi_w(S_w) by interpolating pi_w_table linearily [Pa]
scalar_type pi_w (const scalar_type S_w, const std::vector<scalar_type> pi_w_table,
     const std::vector<scalar_type> S_w_table);

//-------------------------------------------------------------------------------
// Dirichlet boundary condition for pressure (i.e. linear w.r.t the depth). This
    condition is imposed at the top boundaries
scalar_type Pressure(const base_node &x, const scalar_type depth);

//Updates the aperture of the fracture
scalar_type Aperture(const scalar_type bi, const scalar_type bm, const scalar_type
    StrEff,const scalar_type StrEff_i, const scalar_type d);

std::vector<scalar_type> Aperture(const scalar_type bi, const scalar_type bm, const
     std::vector<scalar_type> &StrEff, const std::vector<scalar_type> &StrEff_i,
```

```
    const scalar_type betaf);

std::vector<scalar_type> Aperture(const std::vector<scalar_type> &bi, const std::
    vector<scalar_type> &bm, const std::vector<scalar_type> &StrEff, const std::
    vector<scalar_type> &StrEff_i, const scalar_type betaf);

//Posority correction for the fracture
std::vector<scalar_type> F_phi(const scalar_type bi, const std::vector<scalar_type>
     &b);

std::vector<scalar_type> F_phi(const std::vector<scalar_type> &bi, const std::
    vector<scalar_type> &b);

//Porosity update
std::vector<scalar_type> PhiUp(const scalar_type bi, const std::vector<scalar_type>
     &b, const std::vector<scalar_type> &phi);
```

4.4 The input file

The *GetPot* input file managed by the user is divided in sections and subsections regarding the medium, fracture and time loop data. Beside those there are few extra parameters. The most meaningful parameters that the user is allowed to set are

- The physical dimensions of the domain, and the initial depth of its top. The information on the depth is used for the computation of the hydrostatic boundary condition for pressure, the boundary condition for stress, and possibly the temperature, which could be used in the Arrhenius law for the reaction rates;
- The position of the fracture managed with the LevelSet method
- The fracture initial and maximum thickness and initial porosity
- The maximum number of fixed point iterations and the tolerance of the fixed point iterations
- The sedimentation velocity, which is used at each time step to compute the depth of the source rock;
- The permeability coefficient k_0 and the initial porosity of the non-compacted configuration. These parameters may change a lot from case to case;
- The coefficient in the porosity law β;
- The time step and the number of time iterations
- The resolution of the meshes for the medium and fracture and their type, according to *GetFEM++* classification of the geometric transformation from the reference element to the element of the mesh;
- The integration methods and the finite elements (for velocity, pressure, stress);
- The two parameters for the penalization of boundary conditions and for the stabilization of `problemZpde`.

- Flags and settings for the *Triangle* interface

Here is an example of input file.

```
#GETPOT INPUT FILE

meshFile = meshes/mesh
folderVTK = ./vtk/
stabilize = 1

numberFractures = 1

# Fixed Point Iterations parameters
N_FIXED_POINT_MAX = 100;           # [-]
TOL = 1E-7;                        # [-]

# SUPG parameter for the stress problem
delta = 5.E4;                 # [-]

# Sedimentation velocity
SEDIM_VEL = 1.585E-12;           # [m/s] 0.5 km / 10 My = 500 m / (10 *
    10^6 * 3600 * 24 * 365 s )

# Geothermal gradient of temperature
GRAD_T = 0.035;                    # [C/m]

# Permeability coefficient
K0 = 1.E-6;                        # [Darcy]

# Initial porosity of the non-compacted configuration
PHI0 = 0.5;                        # [-]

# Stress coefficient
BETA = 1.E-8;                      # [Pa^-1]


[dataTime]

        # The timestep and the number of time iterations
        deltaTime = 1000000000000  # [s]
        plotAt = 1
        N_ITER = 3500;                   # [-]

[../]

[stabilization]

    relaxPeclet = 1.
    function = Pe-1.

[../]

[mediumData]

        [./domain]
```

```
                meshExternal = none
                meshFolder = ./meshes/

                triangle = ./Triangle/triangle
                triangleFlags = -pqDa5
                triangleFile = Triangle/FIX.poly

                spatialDiscretizationX = 50
                spatialDiscretizationZ = 30
                DEPTH = 2000. # Initial depth of the top of the domain [
                    m]
                spatialInclination = 0.
                lengthAbscissa = 200.0
                lengthOrdinate = 120.0
                lengthQuota = 1.
                meshType = GT_PK(2,1)
                spaceDimension = 2.

                integrationTypeVelocity = IM_TRIANGLE(6)
                integrationTypePressure = IM_TRIANGLE(1)
                INTEGRATIONSTR = IM_TRIANGLE(2)

                POLYNOMIAL_DEGREE = 1
                FEMTypeVelocity = FEM_RT0(2)
                FEMTypePressure = FEM_PK(2,0)
                FEM_TYPE_STR = FEM_PK(2,1)

                penaltyParameterVelocity = 5.E6
                penaltyParameterPressure = 10.

                nu = 0.25 #Poisson coeff

        [../]

        [./darcy]

                solution = 0 #function for BC on pressure
                solutionIn = 0
                solutionOut = 0
                velocity = 0. #function for BC on velocity

        [../]

[../]
[fractureData0]

        spaceDimension = 1.

        [./levelSet]
                levelSet = y-60.   %the zero is the fracture
                levelSetCut = -1
                zMap = 60  %fracture equation as z=z(x)
```

```
                jacMap = 1. %ratio between grid size and actual arc
                    length
                normalMap = [0.,1.] #[-0.01,1.] #[-0.75,1.] #[-0.25,1.]

                integrationTypeSimplex = IM_STRUCTURED_COMPOSITE(
                    IM_TRIANGLE(3),1)

        [../]

        [./domain]

                position = 0.0
                thickness = 0.01 #initial thickness of the fracture
                thicknessMax = 0.01 #max thickness of the fracture

                phiF0 = 0.9 #initial porosity of the fracture

                                betaF = 1.E-9;    # [Pa^-1] fracture
                                    compressibility

                csi0 = 0.25  # parameter for model closure

                spacing = x

                spatialDiscretization = 160
                translateAbscissa = 0.0
                lengthAbscissa = 200
                lengthOrdinate = 0.
                lengthQuota = 0.
                meshType = GT_PK(1,1)

                integrationTypeVelocity = IM_GAUSS1D(3)
                integrationTypePressure = IM_GAUSS1D(2)

                FEMTypeVelocity = FEM_PK(1,1)
                FEMTypePressure = FEM_PK(1,0)
                FEMTypeLinear = FEM_PK(1,1)

        [../]

        [./darcy]

                solution = 0.0  #for BC on pressure
                velocity = 0.

        [../]


[../]
```

4.5 The interface with an external mesh generator

Moreover the compacted mesh could show significant deformation of the elements even if we start from a mesh of good quality, especially in presence of heterogeneities or large amounts of reactive rock as shown in figures 8 and 9. Since the differential problems are solved on the compacted mesh, in anticipation to the further developments of the project, an effort was made in order to improve the quality of the mesh of the fixed domain.

Because of the limitations of the mesh generator of *GetFEM++*, that is able to mesh just

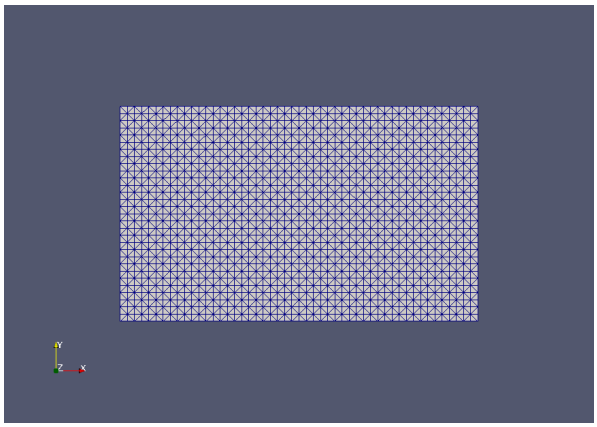

Figure 8: *The initial mesh*

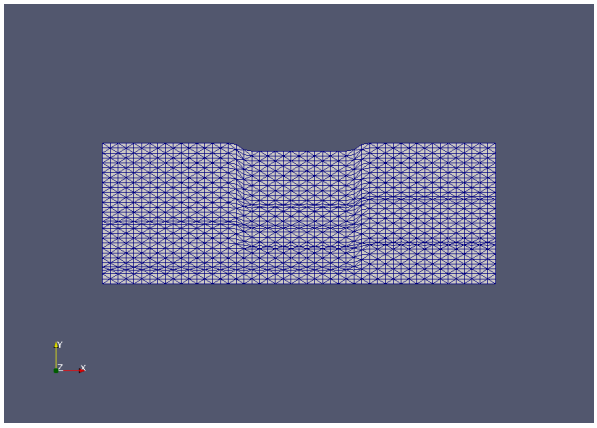

Figure 9: *The mesh of the compacted domain, in the presence of four strips of kerogen partially overlapping, generated with* GetFEM++

rectangular domains, the program has been interfaced with an external mesh generator. So, if it is necessary, the mesh can be generated by *Triangle* that is able to build, for example, Delaunay triangulations, constrained Delaunay triangulations, conforming Delaunay triangulations, (an example is shown in figure 10). All the flags and the options for the use of *Triangle* are selectable directly from the input file. *Triangle* takes as an input

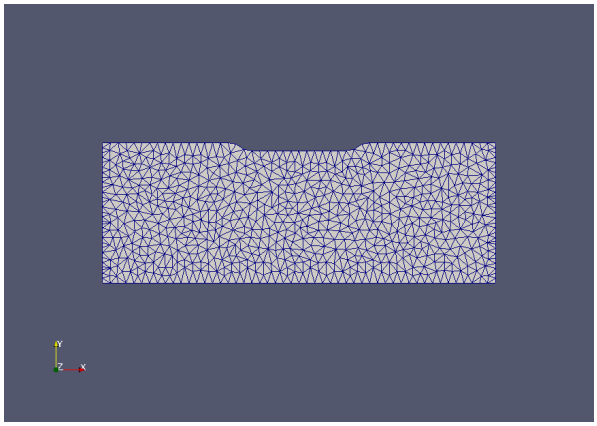

Figure 10: *The mesh of the compacted domain generated with* Triangle

the list of nodes and segments that determine the border of the domain and it meshes the convex hull defined by the border.
To do so we compact the initial mesh, supply the border of the compacted domain to *Triangle* in a *.poly* format, generate the new compacted mesh and convert it to a *.msh* format readable by the *GetFEM++* utilities. Finally we de-compact it and the de-compacted mesh substitute the initial one to keep the order of the nodes between the fixed and the moving mesh, property that is exploited, for example, to visualize the solution on the physical domain. Everything is done by the `MeshHandlerX` with the methods `triangle`, `computePhysPos` and `uncompactMesh`.

5 Results

In this chapter we present the results of the simulations which, as we anticipated, are performed in a two dimensional section of the sedimentary layer. First of all we consider the test case of a fracture that cuts the layer horizontally, studying the influence of its compressibility and considering different boundary conditions.
Once evaluated the validity of the model and the solver with the test case, we move on to the more complex, and more interesting case of inclined fracture that cuts the domain from the top to the bottom.
We consider, as anticipated before, a rock only filled with water and no kerogen ($C(\mathbf{x},t) = C^0 = 0$). Thus no oil can be generated ($S_o(\mathbf{x},t) = S_o^0 = 0$).

5.1 The test case of an horizontal fracture

We studied the case of a fracture that cuts the layer horizontally, analysing its potential impact on the fluid dynamics during the rock compaction. We considered the case of an highly permeable fracture with high initial porosity $\phi_f = 0.9$, $\beta_f = 1.E-6$ Pa^{-1}, initial thickness $l_{\Gamma,i} = 0.01$ m and a maximum thickness $l_{\Gamma,m} = l_{\Gamma,i}$ assuming that the fracture has not experienced previous compaction.
The position of the fracture is located by a level set function at $y = 60m$ and the regular one-dimensional mesh where we solve the reduced problem for the flow in the fracture has a resolution of 160 elements.
The following tests were conducted in a $200m \times 120m$ domain initially at the depth of 2000 m, discretized with a 50×30 regular mesh built with the tools provided by *GetFEM++*. The penalization and stabilization parameters were chosen as $\gamma_U = 5\ 10^6$ (suitably scaled according to the size of the domain) and $\delta = 5\ 10^4$. To take into account compaction we considered a sedimentation velocity of 500 m per 10 My.
The initial conditions for pressure consist of an hydrostatic pressure and the initial conditions for the effective stress and the porosity are obtained with some fixed point iterations: we start from a uniform porosity equal to the porosity of the initial non compacted configuration ($\phi_0 = 0.5$), and compute the overload corresponding to that porosity, obtaining the effective stress as the difference between the overload and the initial pressure in pores. As previously anticipated, the stopping criterion is based on the variations of the Jacobian $J(C, C^0, \phi)$, which accounts for the deformation of the physical domain. Thus, set $\mathbf{J}_k^{n+1}$ the vector of the DOFs of the finite element function $J(C^0, C^0, \phi_k^{n+1})$ we check for each fixed point iteration $k < K_{max}$ if

$$\frac{||\mathbf{J}_k^{n+1} - \mathbf{J}_{k-1}^{n+1}||_2}{||\mathbf{J}_k^{n+1}||_2} < tol.$$

We take a time step $\Delta t = 10^{12}s$ and 3500 time iterations, for a total time span of approximately $111My$.

5.1.1 An horizontal fracture that extends past the boundaries

The boundary conditions selected for the rock medium are homogeneous Neumann conditions on both sides of the domain, which correspond to symmetry conditions, since we assume that the layers continues outside the domain in the horizontal direction. We impose Neumann conditions also at the bottom of the domain since Giovanardi in [12] showed that imposing hydrostatic pressure at the bottom is a strong assumption that can lead to unphysical solutions. Finally we impose Dirichlet conditions at the top, accounting for the hydrostatic pressure. Hence, if we denote with $d > 0$ the depth of the top of the domain and with $z_{top} > 0$ its z coordinate, we require

$$\begin{cases} p = \rho_w \; g \; d & \text{on } \partial_D \hat{\Omega} \\ \mathbf{U} \cdot \mathbf{n} = 0 & \text{on } \partial_N \hat{\Omega} \end{cases}.$$

Regarding the fracture, in this case, we impose symmetry conditions simulating a fracture that extends past the boundaries in the horizontal direction on both sides, i.e. we set

$$\tilde{\mathbf{U}} \cdot \mathbf{n} = 0 \qquad \text{on } \partial\hat{\Gamma}.$$

The flags for the boundary conditions are handled by the `BCHandler` and the pressure values at the Dirichlet boundaries are computed consistently with the user-defined depth. In this test we simulate a progressive sedimentation of the layers above the domain during approximately 100 My. This leads to a progressive burial of the domain, which causes the boundary conditions for both overload and pressure to change.

We show in figure 11 the fixed mesh where we solve the differential problems, along with a close-up of the elements cut by the fracture.

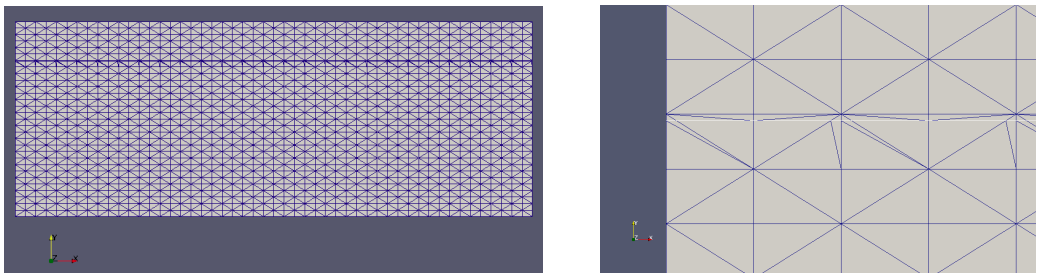

Figure 11: *The fixed mesh and a close-up on the fracture. Notice that, since we are using the XFEM, the re-meshing has the only purpose of visualization.*

The position of the fracture is defined as the zero of the level set function, plotted in figure 12.
In figure 13 we can observe the progressive compaction of the physical domain.
The compaction affects also the porosity of the whole domain as we can see from figure 14, where ϕ decreases significantly from the beginning to the end of the simulation in the whole domain.

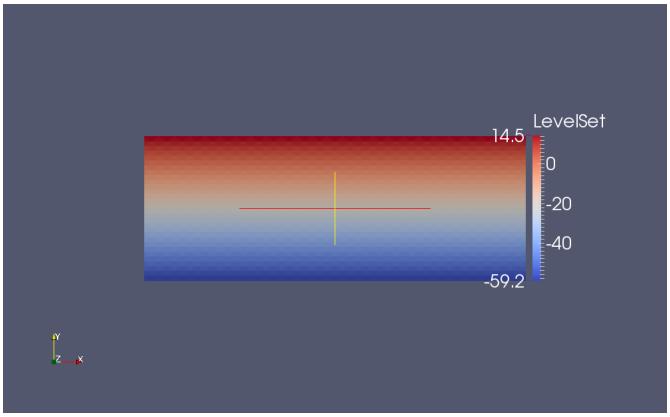

Figure 12: *The level set that defines the position of the fracture.*

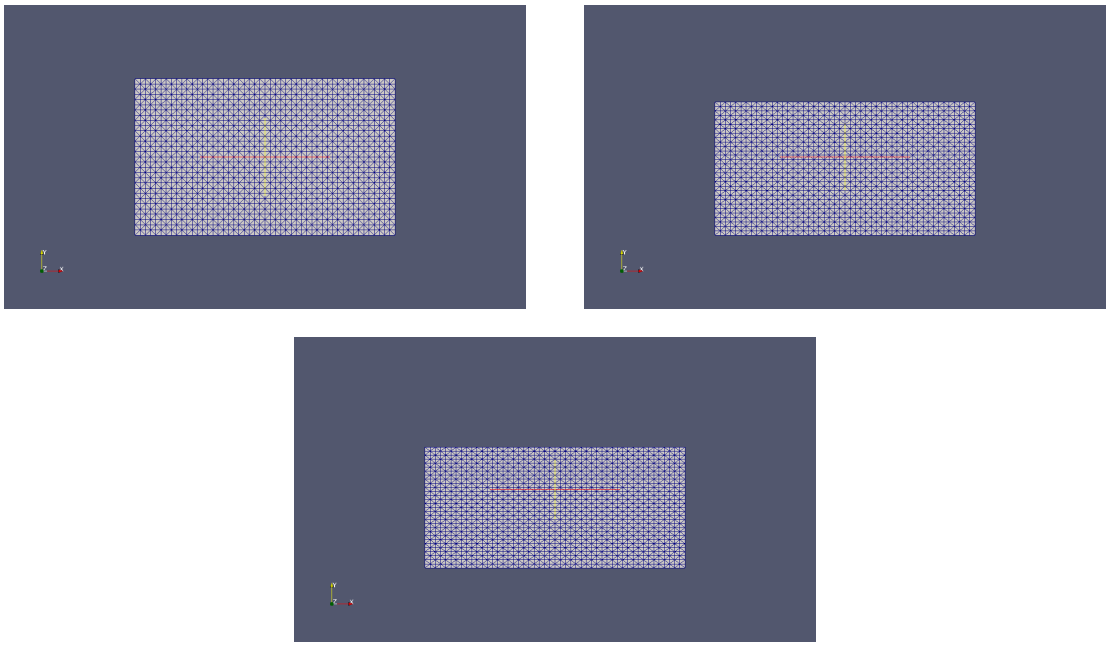

Figure 13: *The deformation of the physical domain. Snapshots taken at* $t = 0,45$ *and* $90\ My$.

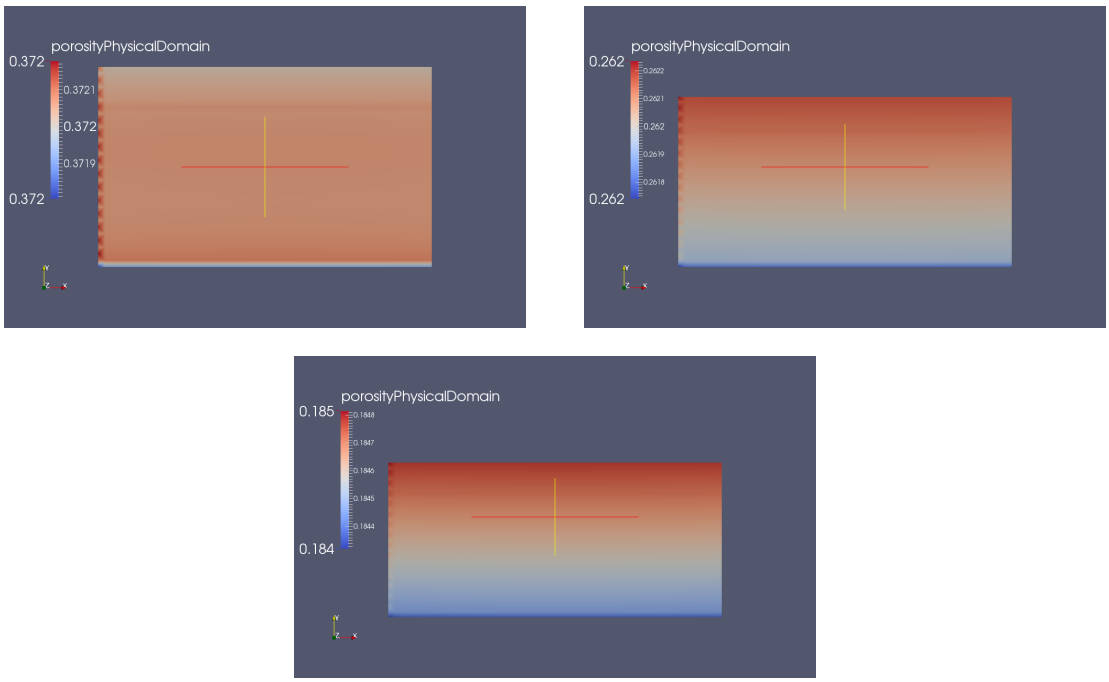

Figure 14: *The evolution of porosity ϕ in the physical domain. Snapshots taken at $t = 0, 45$ and 90 My.*

We can notice few small oscillations, in the left side of the domain, caused by the interpolation of the pressure, defined on the $\mathbb{P}_0$ dof, on the $\mathbb{P}_1$ dof of the stress, performed during the computation of the effective stress.

In figure 15 we report the evolution of the pressure in the medium and fracture. The pressure in both cases increases in time and, as we can see more precisely from the plot with a smaller range in figure 16, the pressure in the fracture is almost homogeneous in space, only slightly higher in the center.

Moreover, looking at the plot of the pressure over a line across the fracture shown in figure 17, we can better see that the pressure in the fracture is comparable to that one that we have in the medium close to it. This is reasonable because we are forcing, with the interface conditions (26), the pressure in the fracture to be equal to the average pressure of the medium across the fracture, plus the jump of the flow, which we expect to be small because of the highly permeable nature of the fracture.

5.1.2 The comparison with the case without fracture

We compared the results of the fractured domain (with $\beta_f = 1.E-6$ Pa^{-1}) to the case with the same settings, but without fracture.

As we can see from figure 18 there is not significant change in the pressure behaviour of the medium in the presence of the fracture. This is reasonable because, even if the fracture is highly permeable, the symmetry boundary conditions at the fracture tips do not allow

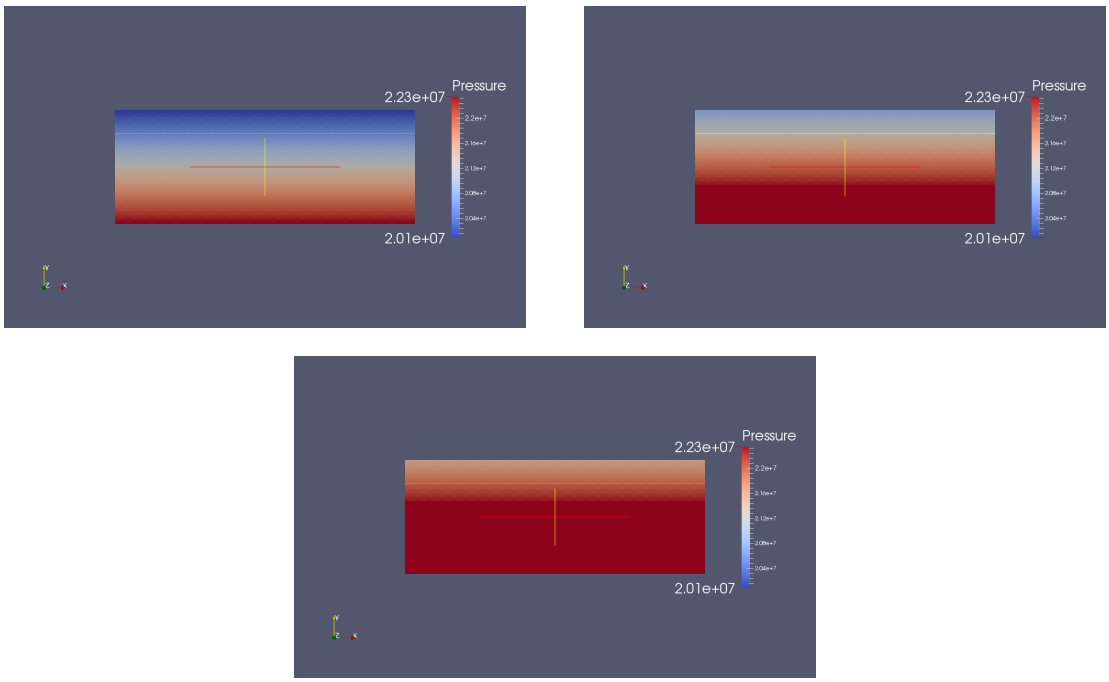

Figure 15: *The evolution of pressure in the rock medium and fracture. Snapshots taken at* $t = 0, 1.5$ *and* 2 My.

fluid inflow/outflow in the tangential direction. Thus an horizontal fracture that extends indefinitely on both sides of the domain does not significantly behave as a preferential path for the flow, and the medium compaction of the domain has a way stronger impact on the overall fluid dynamic behaviour compared to the influence of the fracture.
Moreover, since the presence of the fracture is neglected in the stress problem, as explained in section 2.3, we do not expect significant changes in the effective stress distribution, that ultimately means that is reasonable to not expect much variation in the porosity distribution in the medium. This is confirmed by the plot in figure 19

5.1.3 The impact of fracture compressibility

We performed a parametric study of the behaviour of the fractured domain with different fracture compressibility β_f. Here we report the most meaningful results, regarding the comparison of the cases with $\beta_f = 1.E-6$ Pa^{-1} and with $\beta_f = 1.E-8$ Pa^{-1}.
In the case of $\beta_f = 1.E-8$ Pa^{-1} we have a more significant reduction of the thickness of the fracture because of the compaction. After approximately 90 My the width of the fracture has reduced by roughly the 30%, differently from the other case where we have a reduction lower than 0.1% after 90 My. This implies, in the case of $1.E-8$ Pa^{-1}, a more tangible decrease of the fracture porosity in time, meaning that the fracture tends to lose its high permeability. Moreover, recalling that $\mathcal{K}(\phi) \sim \phi^3$, the permeability is highly sensitive to porosity changes. The progressive reduction of the aperture leads in

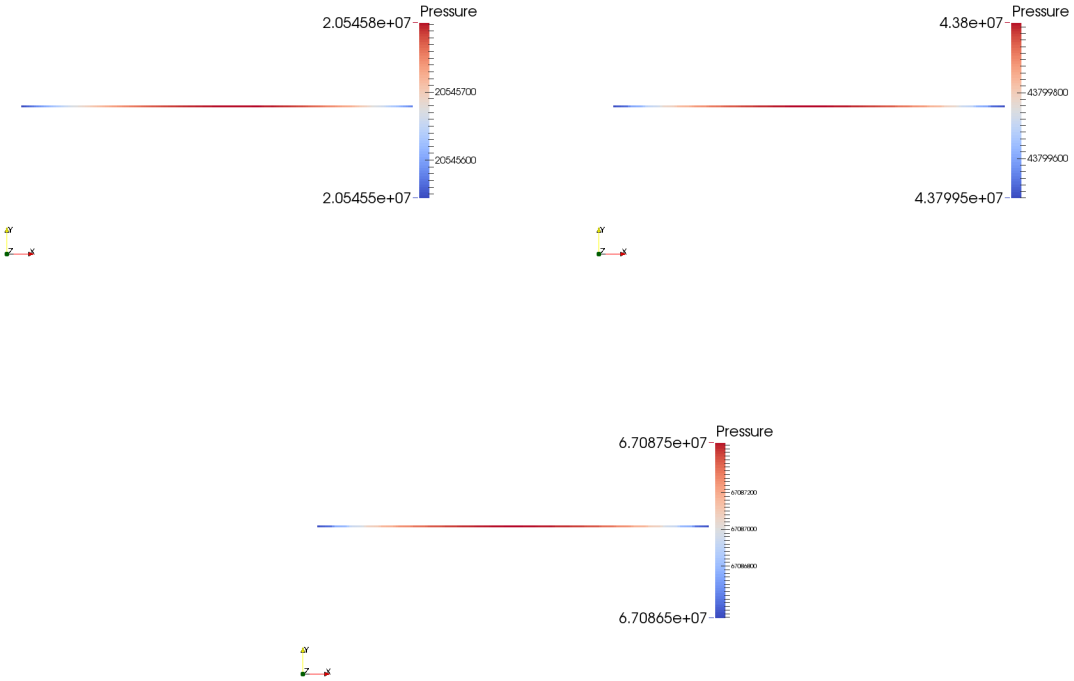

Figure 16: *The evolution of pressure in the fracture. Snapshots taken at* $t = 0, 45$ *and* 90 My.

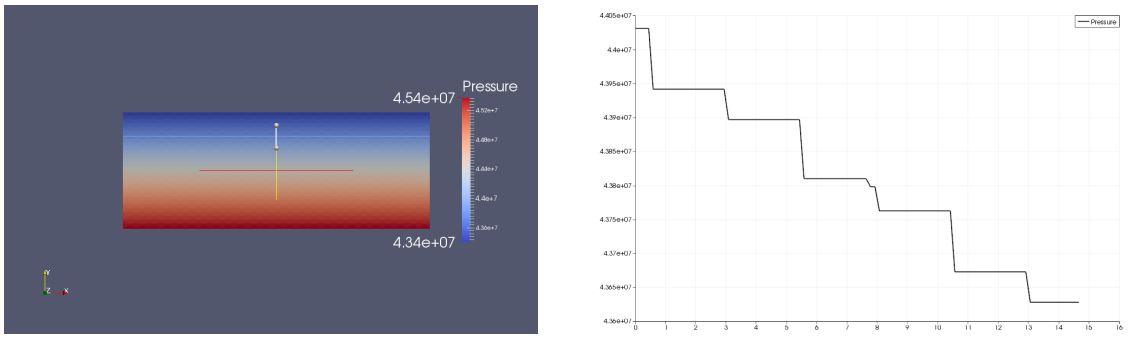

Figure 17: *On the right the plot of the pressure over the line highlighted on the left. Snapshots taken* $t = 45$ My.

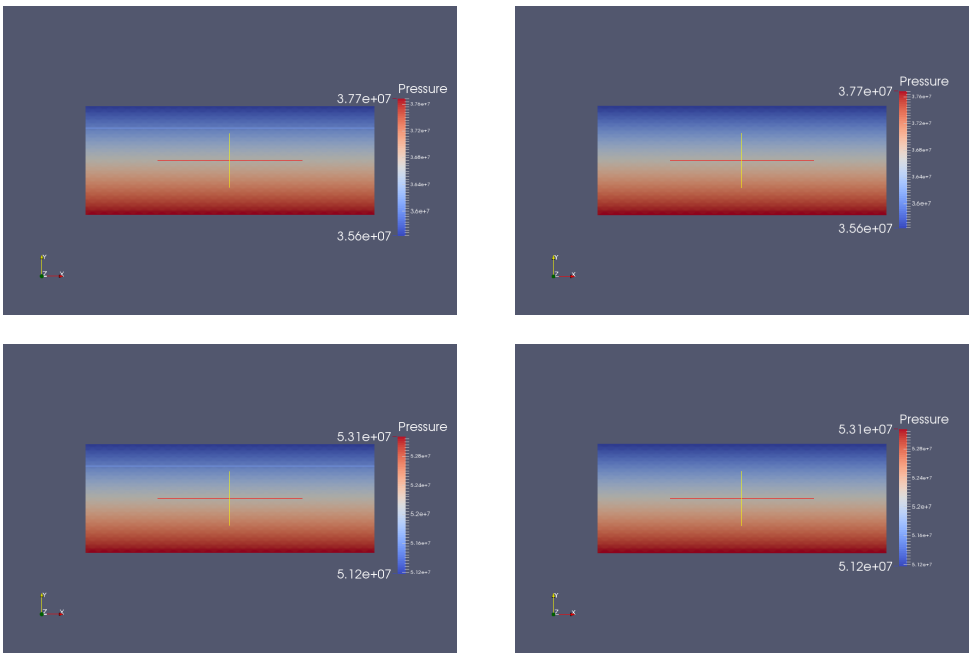

Figure 18: *The evolution of the pressure, on the left the case of the fractured domain, on the right the case with no fracture. Snapshots taken at* $t = 30, 60$ *My.*

general to an higher overpressure, which is the pressure component that does not include the hydrostatic pressure, especially inside the fracture as we can see from figure 20.

Even if the percent variation of the pressure is small, the differences of the pressure values related to the different compressibility are more significant compared to the variation of the pressure that we have in space inside the fracture.
The outcome of this analysis let us presume that the mechanical behaviour of the fracture can have a significant impact in the overall fluid dynamic behaviour of the rock medium. In particular we expect, going toward the limit case of the closed fracture, the formation of two separated compartment of pressure, relative to the domain $\hat{\Omega}_1$ and $\hat{\Omega}_2$.

5.1.4 An horizontal fracture limited to the domain

Differently from the previous case, here we present the results regarding the case of a fracture whose extension is limited to the computational domain. Hence we impose different boundary conditions at the tips with respect to the previous case, in particular the hydrostatic pressure on both ends of the fracture.
Moreover we still consider a rock medium that continues outside the domain in the horizontal direction.
So we require the following boundary conditions for the problem

$$\begin{cases} p = \rho_w \ g \ d & \text{on } \partial_D \hat{\Omega} \\ \mathbf{U} \cdot \mathbf{n} = 0 & \text{on } \partial_N \hat{\Omega} \end{cases}$$

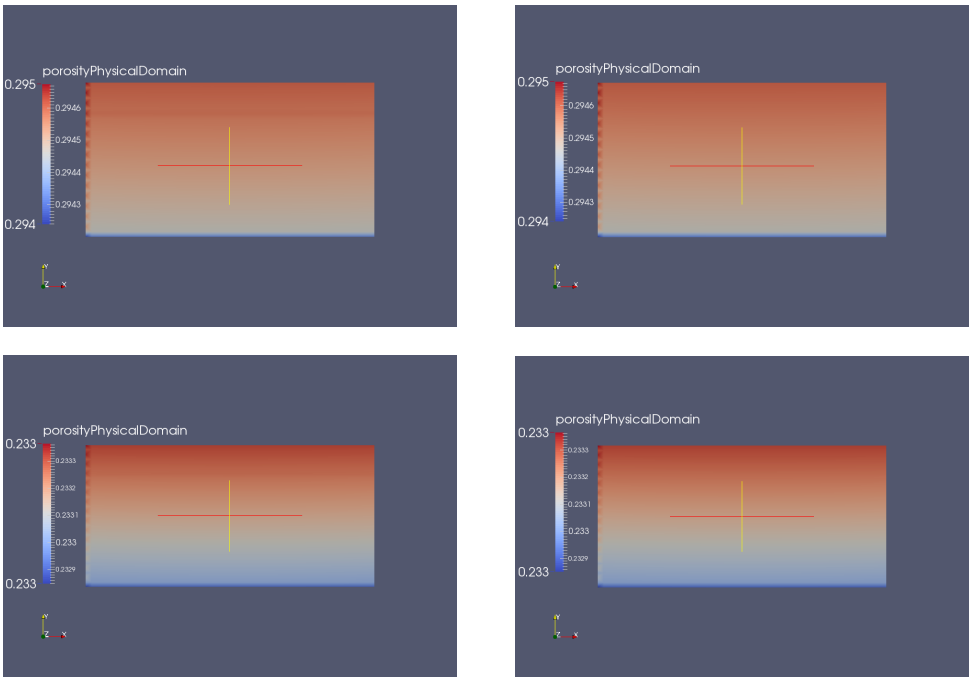

Figure 19: *The evolution of the porosity on the physical domain, on the left the case of the fractured domain, on the right the case with no fracture. Snapshots taken at* $t = 30, 60$ *My.*

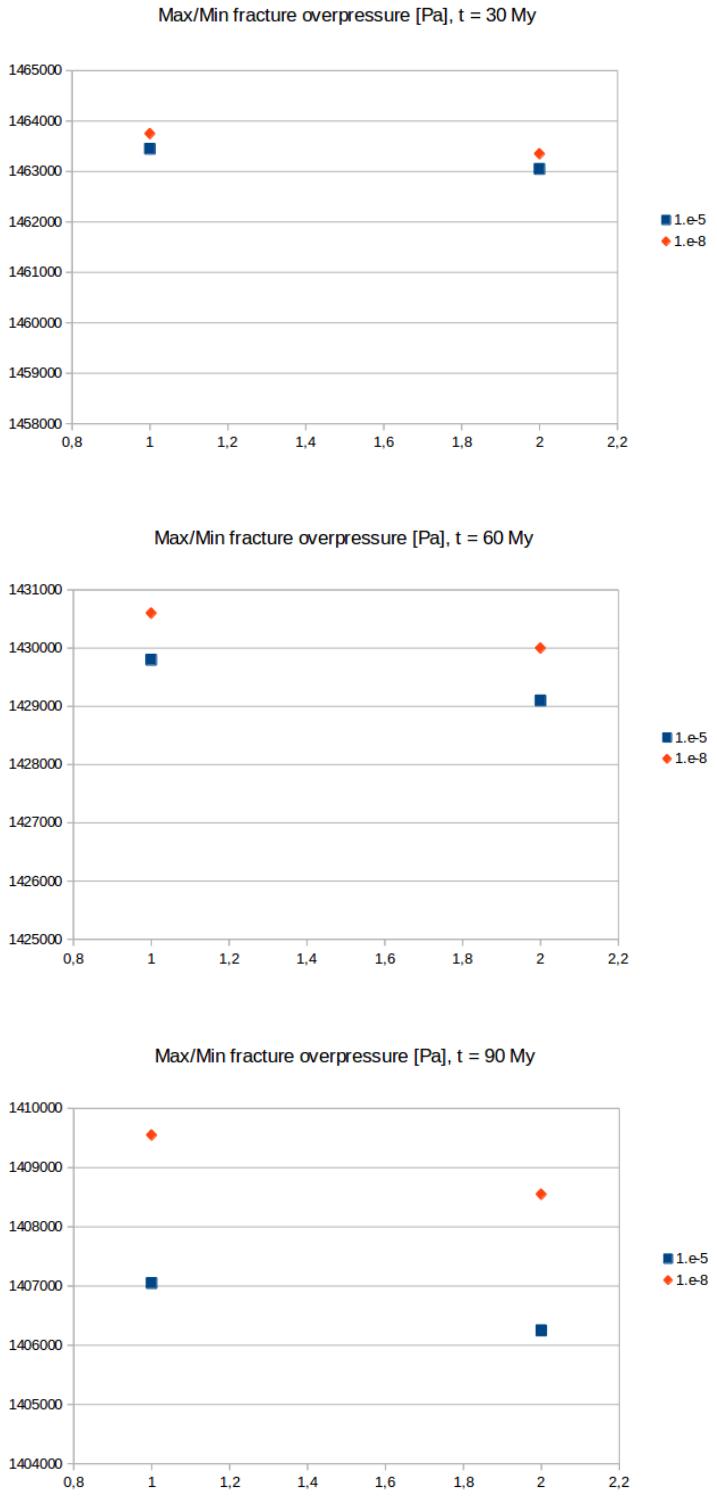

Figure 20: *The Max/Min pressure in the fracture in the case of different compressibility, at* $t = 30, 60, 90$ *My.*

and

$$\tilde{p} = \rho_w \; g \; d_f \qquad \text{on} \;\; \partial\hat{\Gamma},$$

where we denoted with $d_f > 0$ the depth of the fracture.
In this case we have a significant variation in space of the pressure inside the fracture, as shown in figure 21, showing lower values toward both ends of the fracture, compared to the previous case, because of the boundary conditions on the pressure that allow the fluid to flow out from it.

Thus the fracture, in this case, behaves as a vent for the pressure in the domain and its

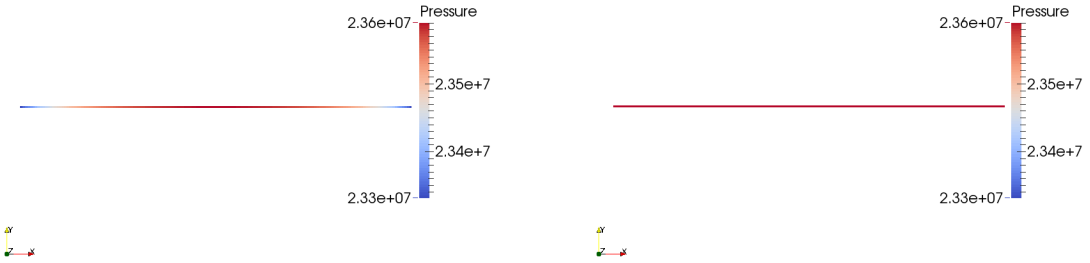

Figure 21: *The pressure, plotted with the same range, in the case of limited fracture on the left and extending fracture on the right. Snapshots taken at* $t = 45 \; My$.

behaviour leads, consequently, to a lower pressure in the rock medium. The lower pressure

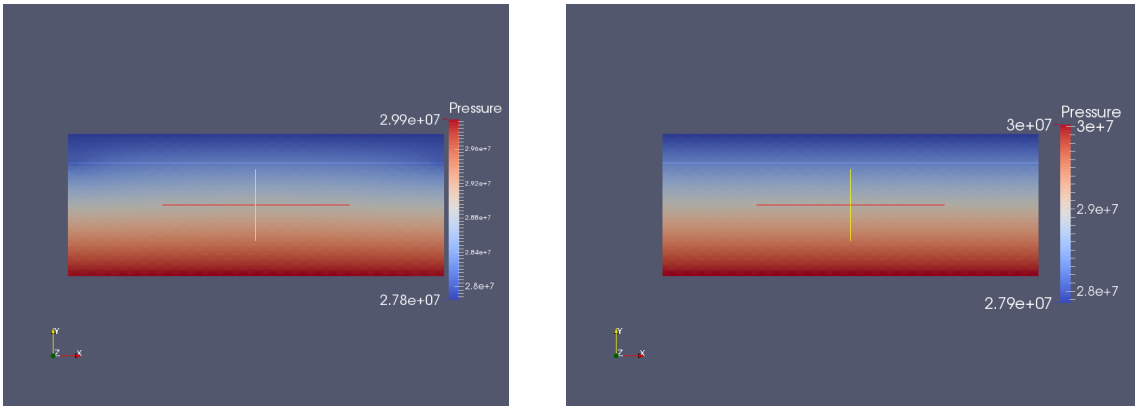

Figure 22: *The Pressure in the full domain in the case of limited fracture on the left and extending fracture on the right. Snapshots taken at* $t = 20 \; My$.

toward the ends of the fracture impacts also the effective stress, that present higher values close to the ends of the fracture, as shown in figure 23. This pattern is obviously followed

by the porosity, represented in figure 24.

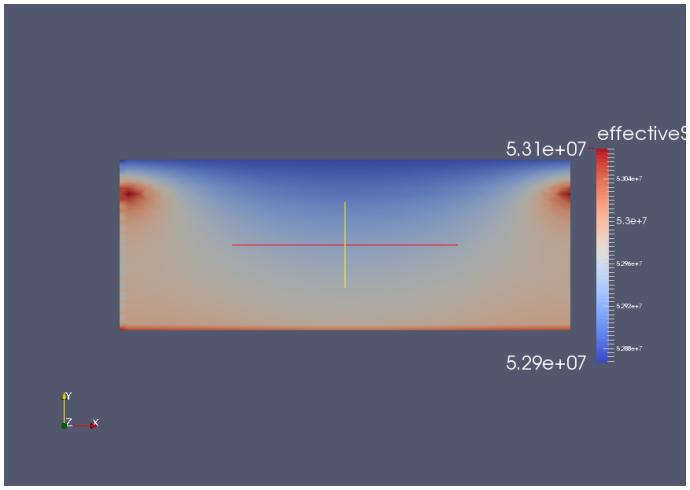

Figure 23: *The effective stress at* $t = 30\ My$.

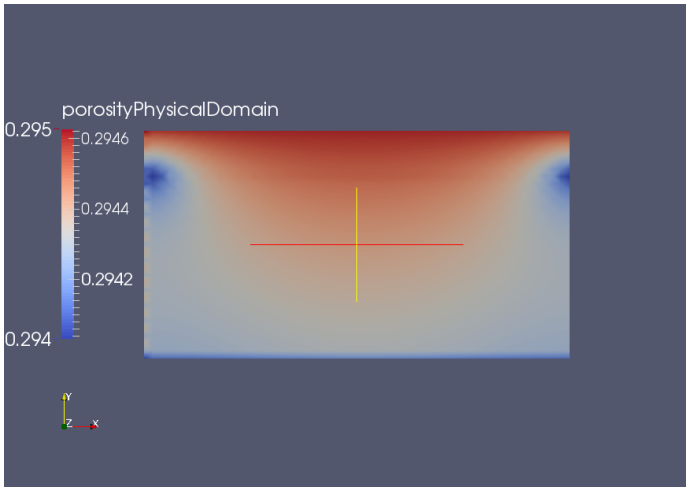

Figure 24: *The porosity plotted in the physical domain at* $t = 30\ My$.

However the high effective stress, in the regions toward the ends of the fracture, determines an higher compression of the fracture in those regions, that ultimately leads to a reduction of the permeability in the long period, reducing the fluid flow and increasing the pressure in the fracture. Thus after 90 My the pressure returns basically homogeneous in the fracture, as shown in figure 25, and the overall distribution of pressure in the domain of the case of the extending fracture is restored, see figure 26.

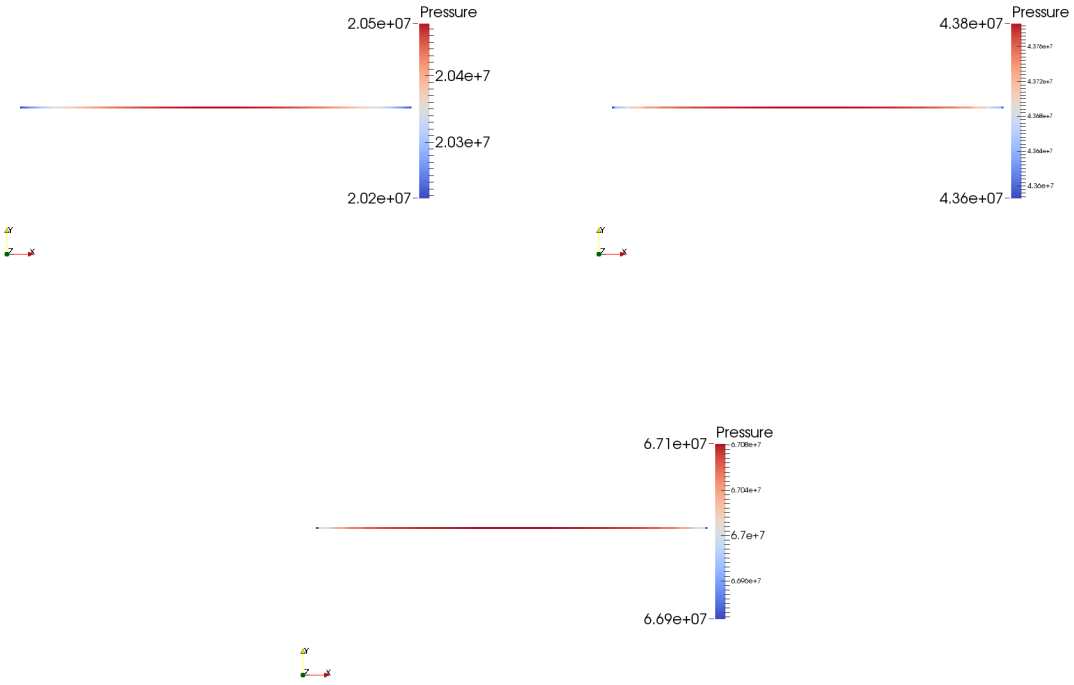

Figure 25: *The evolution of pressure in the fracture. Snapshots taken at* $t = 0, 45$ *and* 90 My.

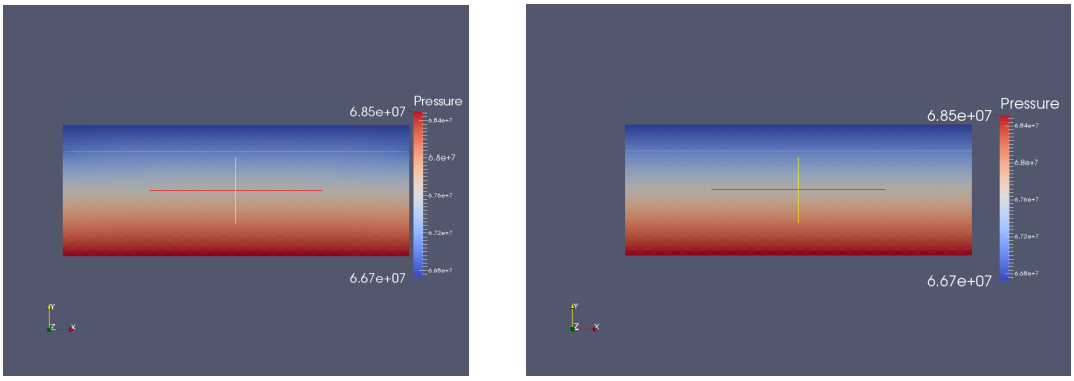

Figure 26: *The pressure in the full domain in the case of limited fracture on the left and extending fracture on the right. Snapshots taken at* $t = 90\ My$.

5.2 Inclined fracture

After testing the model and the solver with the horizontal fracture we moved on with the analysis of an inclined fracture that cuts the domain from the top to the bottom as shown in figure 27. We consider a fracture with an initial porosity of $\phi_f = 0.9$, $\beta_f = 1.E{-}8\ \text{Pa}^{-1}$, initial thickness $l_{\Gamma,i} = 0.01$ m and a maximum thickness $l_{\Gamma,m} = l_{\Gamma,i}$. The other parameters are the same of the previous cases.
Moreover, since the fracture is inclined, in this case the horizontal effective stress plays a role in the fracture compaction. We consider the following Poisson coefficient in (5), $\nu = 0.25$ so that we have $\sigma_{T,H} = \frac{1}{3}\sigma_T$.

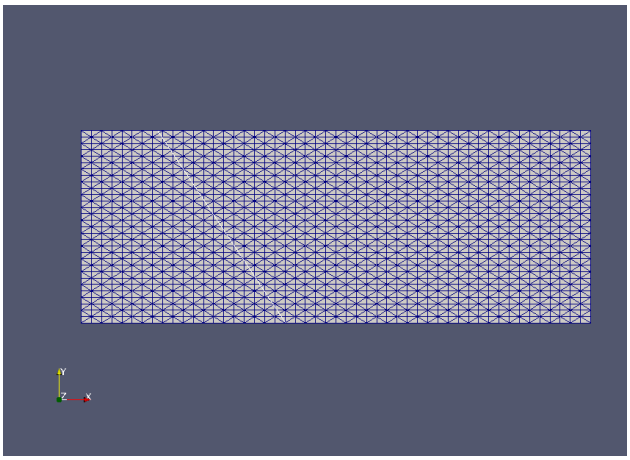

Figure 27: *The position of the fracture in the fixed domain, defined by a line with angular coefficient of -1.5.*

As we did previously, we consider a rock medium that extends outside the domain in the horizontal direction and since the fracture cuts the domain from the top to the bottom

we impose the following boundary conditions

$$\begin{cases} p = \rho_w \ g \ d & \text{on } \partial_D \hat{\Omega} \\ \mathbf{U} \cdot \mathbf{n} = 0 & \text{on } \partial_N \hat{\Omega} \end{cases}$$

and

$$\begin{cases} \tilde{p} = \rho_w \ g \ d & \text{on } \partial \hat{\Gamma}_{top}, \\ \tilde{\mathbf{U}} \cdot \mathbf{n} = 0 & \text{on } \partial \hat{\Gamma}_{bottom}. \end{cases}$$

The presence of the fracture has a meaningful impact on the pressure distribution of the rock medium, as it is noticeable in the left figure 28, where we observe a local drop of the pressure. This phenomenon happens because, thanks to the high permeability of the fracture, the fluid is allowed to flow upwards where the boundary conditions allows for outflow. Thus the fracture acts as a vent for the pressure, draining the fluid from the lower part of the domain to the upper boundary.
This is also confirmed considering the same case but with a fracture with low permeability, with an initial porosity of $\phi_f = 0.4$. Looking at figure 28 we can see that in the case of low permeability the fracture is not able to become a preferential path for the fluid flow and consequentially it does not determine a drop of pressure.

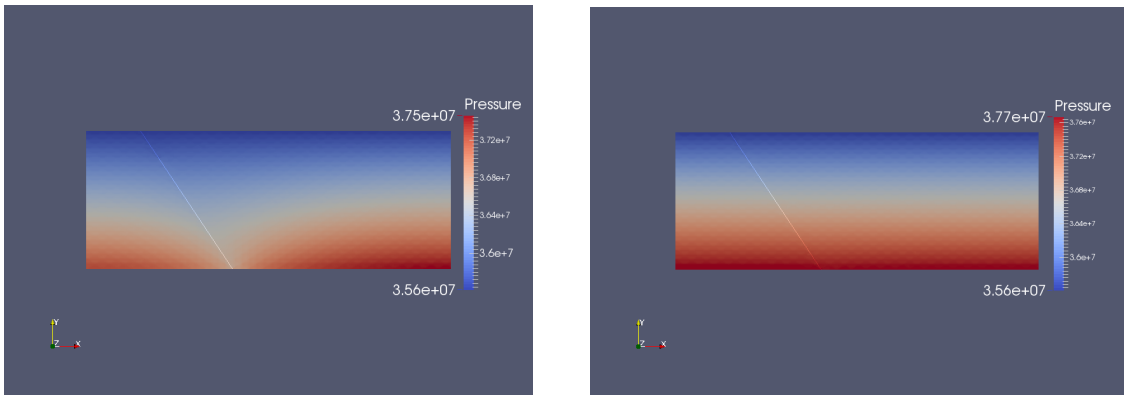

Figure 28: *The pressure distribution in the fixed domain in the case of inclined fracture after* 30 My. *Fracture with high permeability on the left and low permeability on the right.*

The pressure asymmetry is reflected, obviously, on the effective stress that, as shown in figure 29, shows a region of maximum value localized at the lower end of the fracture.

The uneven distribution of pressure affects the compaction of the domain. In the low-pressure region close to the bottom, the fluid is, indeed, not able to sustain the load, causing the presence of a minimum porosity region, as figure 30 shows. Thus the region close to the base of the fracture is more likely to collapse compared to the rest of the domain as attested by the small depression at the top of the domain localized in correspondence of the fracture lower end, as it is noticeable from the zoom in figure 31.

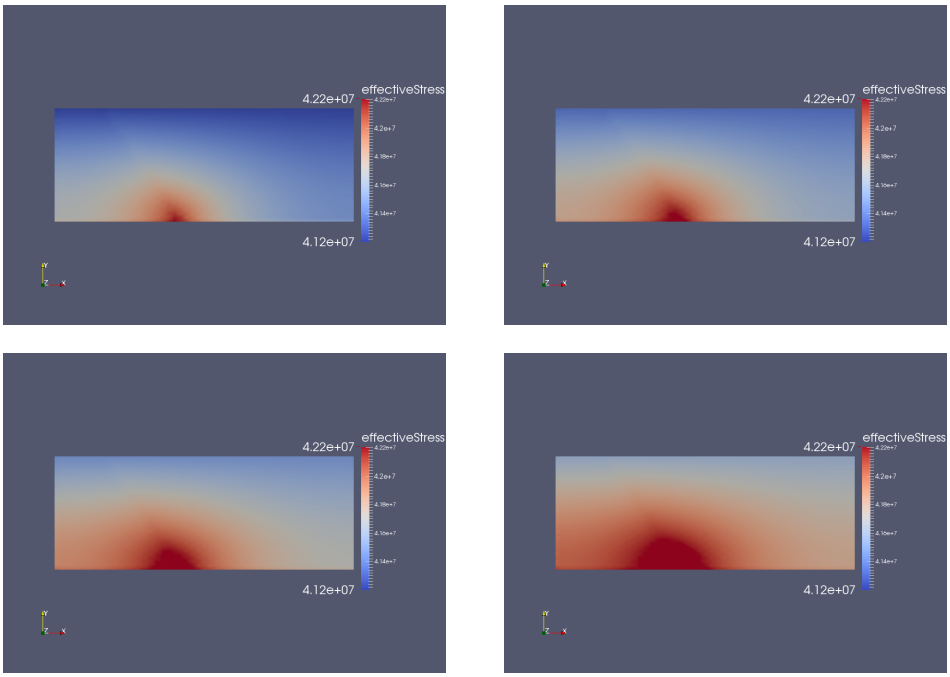

Figure 29: *The evolution of the effective stress, after 500, 505, 510 and 515 time iterations, showing a region of high values at the bottom of the fracture.*

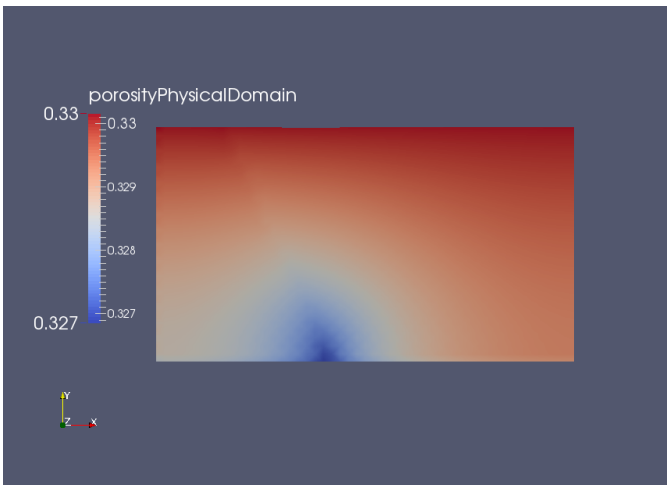

Figure 30: *The porosity plotted in the physical domain in in the case of inclined fracture after* 20 My.

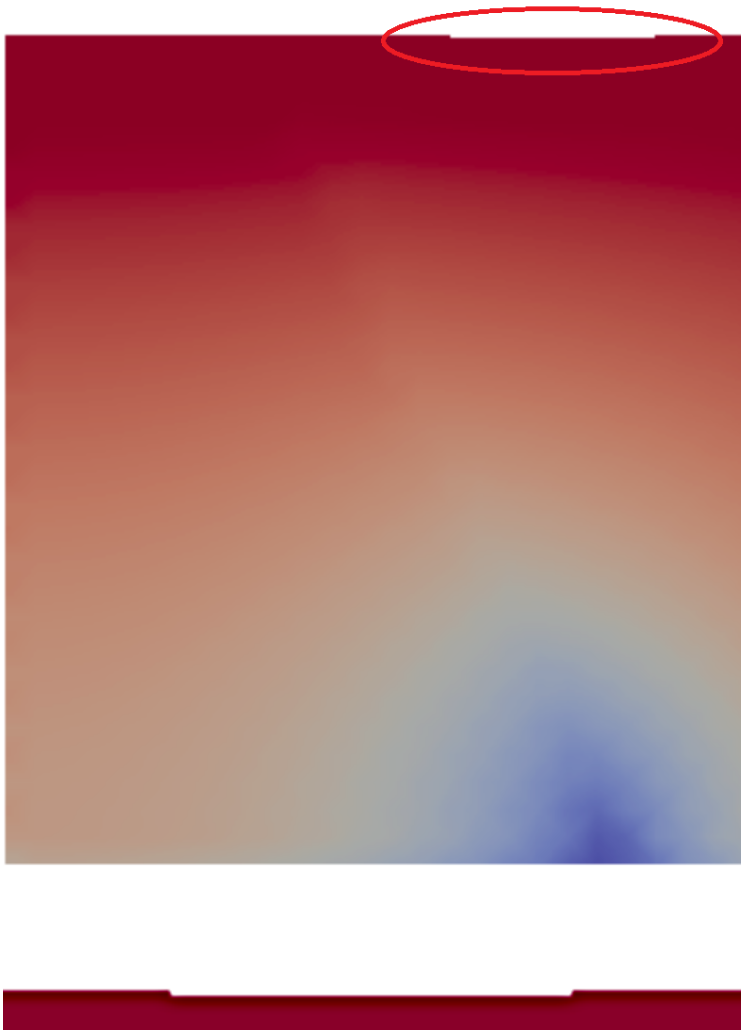

Figure 31: *Zoom on the top of the domain where a depression detectable in correspondence of the lower end of the fracture. Snapshot taken at* $t = 20\ My$.

5.2.1 The influence of slope

We also considered different inclinations of the fracture, studying their impact on the fluid dynamic behaviour of the fracture. We report, for instance, the case of higher inclination represented in figure 32. All other physical parameters are the same of the previous inclined case.

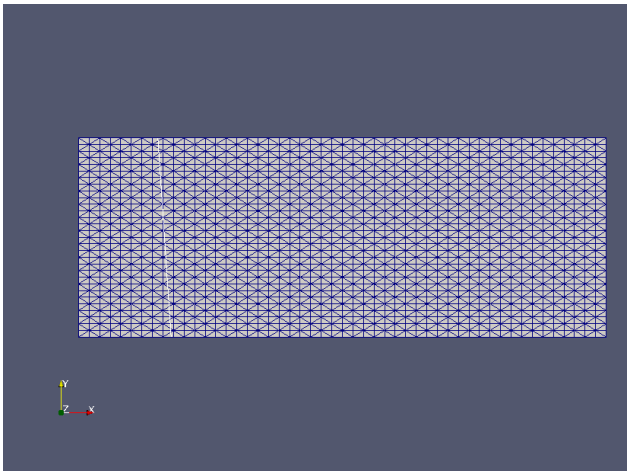

Figure 32: *The position of the fracture in the fixed domain, defined by a line with angular coefficient of -15.*

Because of the higher inclination the horizontal stress plays the major role in the law which models the evolution of the fracture thickness (4). Since the horizontal stress has a smaller entity compared to that one of the vertical stress, it is reasonable to expect that the highly inclined fracture would be subject to a slower compaction process and thus is able to keep its permeability high for a longer time.

So, because of the higher permeability, in the long time frame, we observe a progressive lower overpressure inside the fracture, with respect to the case of lower inclination, as shown in figure 33. We point out that the total pressure increases in time because the predominant component is the hydrostatic pressure, which grows in time because of the burial of the domain. However the overpressure decreases in time because, as the time elapses, we proceed toward an equilibrium between pressure and compaction. Thus the compaction slows down and the motion of the fluid reduces, going toward a still situation of pure hydrostatic pressure.
The different pressure behaviour of the highly inclined fracture ultimately determines a lower minimum porosity value, compared to the case of lower inclination, as represented in figure 34.

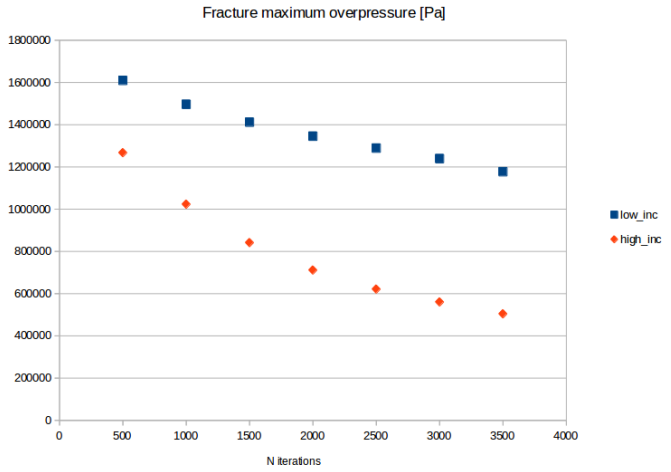

Figure 33: *The maximum overpressure inside the fracture for the case of lower inclination (angular coefficient of -1.5) and higher inclination (angular coefficient of -15).*

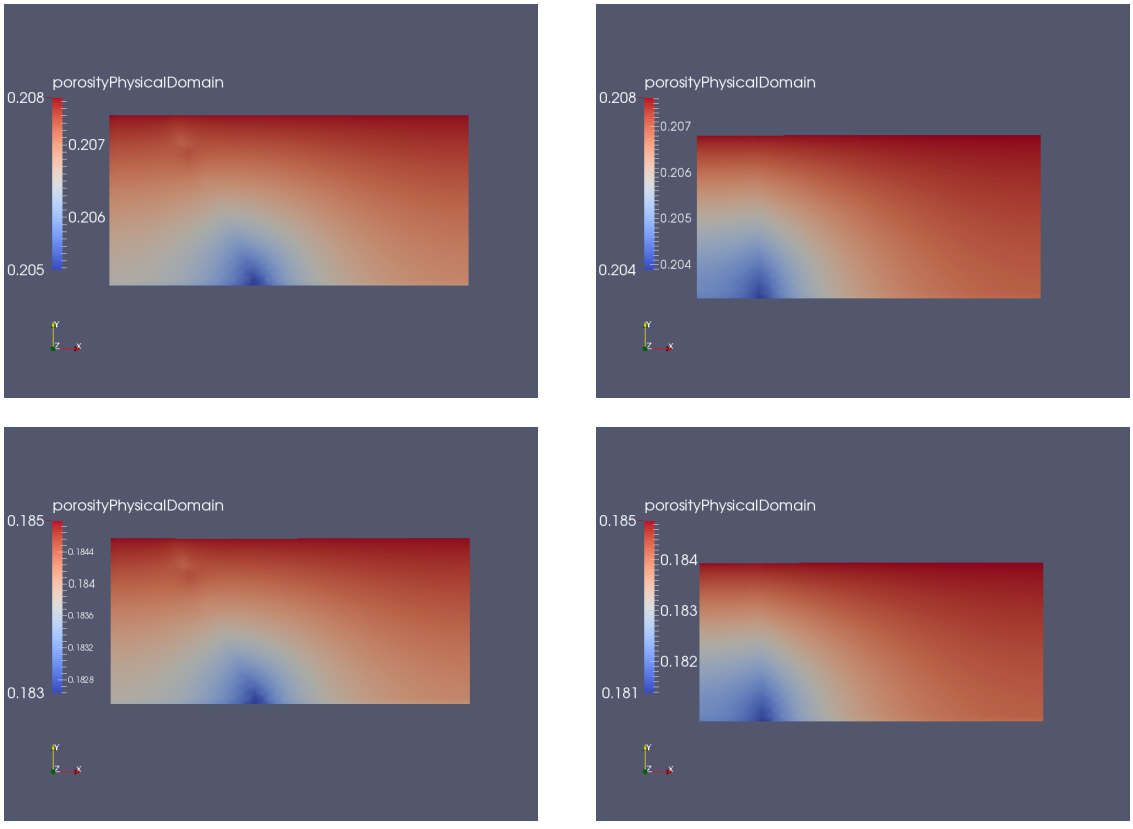

Figure 34: *The porosity plotted in the physical domain in the case of low inclination on the left and high inclination on the right. Snapshots taken* $t = 75$ *and* 90 *My.*

6 Conclusions and further developments

In this work we presented a suitable model able to represent the compaction phenomenon in a fractured sedimentary layer and we built an appropriate solver for the numerical resolution of the resulting coupled, non-linear problem.
The numerical results show a fluid dynamic behaviour consistent with the physics of the problem and, furthermore, the changes of the fracture properties that we analysed determine a reasonable behaviour of the fracture. Finally we showed how sensitive the problem is to the different boundary conditions at the tips of the fracture.
We report, below, the next steps that we consider as the most meaningful in order to extend the project.

- In this work we deeply analysed the influence of a single fracture that cuts the sedimentary layer. An interesting development of the analysis could be considering networks of multiple fractures localized in a portion of the sediment, or fractures that branch out.
  Since those configurations would determine an even more significant local increase in the permeability of that portion of the domain, we would expect a sensible intensification of the draining effect and, consequentially, an even stronger asymmetry in the rock compaction process.

- As previously anticipated, a natural continuation of the work would be taking into account chemical reactions that modify porosity, reducing it in the case of mineral deposition, or locally increasing it when the solid matrix is converted into fluid. The, aim would be the development of a general model able to represent these processes in presence of cracks, modeling, for example, the cementation of fracture due to mineral precipitation.
  Moreover, the generation of hydrocarbons and their primary migration from the source rock, where they are generated, is of great interest for the petroleum industry. Oil is generated from the layers of sediments rich in organic matter, such as kerogen, which, because of the high temperatures experienced by the source rock due to its progressive burial, is subject to chemical reactions that cause its breakdown into oil. The treatment for the fracture compaction, employed in our work, could be leveraged to represent more realistic sources of kerogen that have often the shape of thin layers.

- Describing our model for the mechanical compaction we assumed uniaxial consolidation, which is a reasonable assumption since the deformation in the other two directions of the sedimentary layer are limited because of the presence of surrounding rock, and we assumed the fracture aperture to be small enough to guarantee the continuity of the stress across the crack. However, the model for the stress evolution could be refined, considering a full elasticity problem, relaxing the hypothesis of uniaxial consolidation and taking into account the presence of the fracture, allowing a potential discontinuous state of stress across it.

References

[1] L.F. Athy. Density, porosity, and compaction of sedimentary rocks. *AAPG Bulletin*, 14(1), 1–24 1930.

[2] R. Becker, P. Hansbo and R. Stenberg. A finite element method for domain decomposition with non-matching grids. *ESAIM: M2AN*, 37:209–225, 2003.

[3] R. Becker, E. Burman and P. Hansbo. A Nitsche extended finite element method for incompressible elasticity with discontinuous modulus of elasticity. *Comput. Methods Appl. Mech. Eng.*, 198:3352–3360, 2009.

[4] O.Borromeo, D. Leythaeuser, F.Mosca, R.di Primio, M.Radke and R.G.Schaefer. Pressure solution in carbonate source rocks and its control on petroleum generation and migration. *Marine and petroleum geology*, 12, no. 7: 717–733, 1995.

[5] R.H. Brooks and A.T. Corey. Hydraulic Properties of Porous Media. *Hydrology Papers, No. 3, Colorado State U., Fort Collins, Colorado.*, 1964.

[6] Z. Chen, R.E. Ewing, H. Lu, S.L. Lyons, S. Maliassov, M.B. Ray, and T. Sun. Integrated two-dimensional modeling of fluid flow and compaction in a sedimentary basin. *Computational Geosciences*, 6:545–564, 2002.

[7] Carlo D'Angelo and Anna Scotti. A mixed finite element method for Darcy flow in fractured porous media with non-matching grids. *ESAIM: Mathematical Modelling and Numerical Analysis*, 46(2):465–489, December 2011.

[8] T. Engelder. Stress Regimes in the Lithosphere. *Princeton University Press*, 1993.

[9] Alessio Fumagalli. Numerical Modelling of Flows in Fractured Porous Media by the XFEM Method. *PhD thesis, Politecnico di Milano*, 2012.

[10] Alessio Fumagalli and Anna Scotti. Numerical modelling of multiphase subsurface flow in the presence of fractures. *Communications in Applied and Industrial Mathematic*, 3(1), 2011.

[11] Alessio Fumagalli and Anna Scotti. A numerical method for two-phase flow in fractured porous media with non-matching grids. *Advances in Water Resources*, 62:454–464, 2013.

[12] Bianca Giovanardi. Numerical modeling of porosity evolution in source rock during kerogen breakdown. *Master thesis, Politecnico di Milano*, 2013.

[13] A. Hansbo and P. Hansbo. An unfitted finite element method, based on Nitsche's method, for elliptic interface problems. *Comput. Methods Appl. Mech. Eng.*, 191:5537–5552, 2002.

[14] V. Martin, J. Jaffré and J.E. Roberts. Modeling fractures and barriers as interfaces for flow in porous media. *SIAM J. Sci. Comput.* 26:1667–1691, 2005.

[15] Yves Renard and Julien Pommier. Getfem++ Short User Documentation. 2013.

[16] J. Rutqvist, Y.-S. Wu, C.-F. Tsang, and G. Bodvarsson. A modeling approach for analysis of coupled multiphase fluid flow, heat transfer, and deformation in fractured porous rock. *International Journal of Rock Mechanics and Mining Sciences*, 39(4):429–442, June 2002.

[17] C Vinci, J Renner, and H Steeb. A hybrid-dimensional approach for an efficient numerical modeling of the hydro-mechanics of fractures. pages 1–54, 2013.

[18] Magnus Wangen. Vertical migration of hydrocarbons modelled with fractional flow theory. (1993):109–131, 2007.