An Advanced Discrete Fracture Methodology for Fast, Robust, and Accurate Simulation of Energy Production from Complex Fracture Networks

S. de Hoop[1], D. V. Voskov[1,2], G. Bertotti[1], and A. Barnhoorn[1]

[1]Delft University of Technology, Delft, the Netherlands
[2]Stanford University, Stanford, United States of America

Key Points:

- We developed an open-source preprocessing tool that can create, at the required level of accuracy, a fully conformal uniformly distributed grid for a given realistic 2D fracture network with variable aperture distributions;
- ~~This leads to a robust way of constructing a hierarchy of DFMs for uncertainty quantification of energy production from reservoirs with natural fracture networks~~;
- We demonstrate the application of the developed tool with complex, realistic fracture networks;
- The proposed methodology allows us to use accurate DFM models with similar computational complexity as the EDFM approach.
- ~~The simplified DFM representations of fracture networks still capture characteristics and flow response of real complex networks;~~
- A novel aperture correction procedure allows us to capture the original high fidelity connectivity and flow patterns;
- We also show that it is necessary to process the raw fracture data for topological analysis.

Corresponding author: S. de Hoop, `S.dehoop-1@tudelft.nl`

Abstract

Fracture networks are abundant in subsurface applications (e.g., geothermal energy production, CO2 sequestration). Fractured reservoirs often have a very complex structure, making modeling modeling of flow and transport in such networks slow and unstable. Consequently, this limits our ability to perform uncertainty quantification and increases development costs and environmental risks. This study provides an advanced methodology for simulation based on Discrete Fracture Model (DFM) approach. ~~Changes to the topology of the fracture network reduce computational complexity while preserving the accuracy of the DFM approximation.~~ The preprocessing framework results in a fully conformal, uniformly distributed grid for realistic 2D fracture networks at a required level of precision. The simplified geometry and topology of the resulting network are compared with input (i.e., unchanged) data to evaluate the preprocessing influence. The resulting mesh-related parameters, such as volume distributions and orthogonality of control volume connections, are analyzed. Furthermore, changes in fluid-flow response related to preprocessing are evaluated using a high-enthalpy two-phase flow geothermal simulator. The simplified topology directly improves meshing results and, consequently, the accuracy and efficiency of numerical simulation. The main novelty of this work is the introduction of an automatic preprocessing framework allowing us to simplify the fracture network down to required level of complexity and addition of a fracture aperture correction capable of handling heterogeneous aperture distributions, low connectivity fracture networks, and sealing fractures. The graph-based framework is fully open-source ~~, based on graph theory, and that simplifies the topology of fractures~~ and explicitly resolves the small-angle intersections within the fracture network. ~~Augmenting the framework with a~~ A rigorous analysis of changes in the static and dynamic impact of the preprocessing algorithm demonstrates that~~, we demonstrate that~~ explicit fracture representation can be computationally efficient, ~~used together with maintaining computational efficiency~~ enabling their use in large-scale uncertainty quantification studies.

Plain Language Summary

Fractured rocks occur naturally and are abundant in the earth's subsurface, especially in rocks that host a variety of resources, from geothermal energy to clean water. Modeling fluid flow in such systems is complex and time-consuming, increasing environmental and economic risks. We attempt to tackle this problem by introducing an advanced modeling technique that simplifies the fractures' representation while maintaining the main characteristics. The method's performance is analyzed based on changes in the geometry of the fractures and fluid flow patterns. The framework manages to significantly speed up the required time for fluid flow calculations while remaining close to the high fidelity solution (i.e., solution of unchanged fracture configuration). Because most of the parameters in subsurface-related energy applications are uncertain, many simulations have to be carried out to quantify these uncertainties. Since our framework reduces the computational time, more simulations could be executed, reducing the risks associated with the development of subsurface energy resources.

1 Introduction

Many subsurface energy applications (e.g., geothermal energy production) rely on accurate numerical simulations of fluid flow and mass or heat transport in fractured porous media. A large class of methods is available for numerical modeling of fracture networks. It may consist of various approaches to the homogenization of fractures network, including Dual Porosity (Barenblatt, 1960; Warren & Root, 1963) and various MINC models (Pruess & Narasimhan, 1982; Karimi-Fard et al., 2006), or different versions of Embedded Discrete Fracture Models (EDFM) starting from already classic approaches (L. Li & Lee, 2008; Hajibeygi et al., 2011) to projection-based technique (Ţene et al., 2017; HosseiniMehr et al., 2020). Some hybrid versions combining EDFM with homogenized fractured networks at two different scales ~~are~~ also exist (L. Li & Voskov, 2021).

Another class of model is Discrete Fracture Model (DFM), where fracture segments are described as a lower-dimensional object on the mesh. The ideas of the modern DFM approach can be found in Gureghian (1975) where Finite Element Methods (FEM) was applied, and in Helmig et al. (1997) where Finite Volume Method (FVM) was used. In modern reservoir simulation, which includes highly implicit time approximation using finite-volume discretization on unstructured grids, the DFM methodology has been introduced by Karimi-Fard et al. (2004). The DFM approach is often preferred in detailed geological studies due to its accuracy (Moinfar et al., 2011; Flemisch et al., 2018; Berre et al., 2019; Wong et al., 2020). DFM models typically require a high meshing accuracy to resolve the fracture networks' complex geometry, thereby drastically increasing the computational complexity and rendering them unusable for uncertainty quantification purposes (Jung et al., 2013; Nejadi et al., 2017; Spooner et al., 2021). More recently, the DFM approach in reservoir simulation has been enhanced for practical applications by fully coupling geomechanics (Garipov et al., 2016) and fracture propagation (Gallyamov et al., 2018). These complex physical processes typically require a fine modeling resolution to capture all the effects, further exposing the limitations of incorporating uncertainty quantification.

These limitations severely constrain~~t~~ the necessary low-risk, sustainable, and energy-efficient subsurface activities that are desired. One of the main factors of the considerable computational complexity of ~~the~~ DFM models is the meshing artifacts (i.e., skinny triangles, small control volume sizes, and a large number of degrees of freedom) that result from using conformal meshes and related convergence issues (Geiger & Matthäi, 2014; X. Li & Li, 2019; Koohbor et al., 2020). Fracture network input data is typically acquired from outcrop analysis or statistical models. In outcrop analysis, raw output, either by manual or automatic interpretation, results in difficulties for the meshing software. These meshing artifacts are highlighted in Figure 1 and are well known in the existing literature (Reichenberger et al., 2006; Mustapha & Mustapha, 2007; Mallison et al., 2010; Karimi-Fard & Durlofsky, 2016; Berre et al., 2019).

Several preprocessing strategies have been proposed in the literature to address the challenges of constructing a conformal mesh for complex natural fracture networks. However, the investigation of a numerically convergent solution after applying the preprocessing procedure, a thorough examination of the topology changes as a function of discretization accuracy, and the application to uncertainty quantification have not been adequately studied. Furthermore, in most existing methods, the meshing challenges related to fracture segments intersecting at a small angle are only implicitly resolved. For example, in most studies, an algebraic constraint is used for merging nodes, but the angle at which fractures intersect is not explicitly checked. This means that some meshing issues are not resolved. Finally, in the existing fracture preprocessing methods, variability of the fracture aperture is not taken into account.

Therefore, we have developed an open-source preprocessing framework that borrows concepts from early work in this area (Koudina et al., 1998; Maryška et al., 2005) and more recent approaches (Mustapha & Mustapha, 2007; Mallison et al., 2010; Karimi-Fard & Durlofsky, 2016). It differs from other graph simplification works, such as Wellman et al. (2009), where small (low permeable) fractures are iteratively removed. According to prescribed algebraic constraints, our preprocessing procedure merges nodes and resolves fractures that intersect at a significantly small angle that would otherwise introduce additional meshing challenges. To capture variable aperture distribution and low connectivity networks, in addition to previous methodologies, an aperture correction is added to the method presented here. Most of the operations are formulated using graph theory, which results in simple bookkeeping of the incidence matrix operations (West et

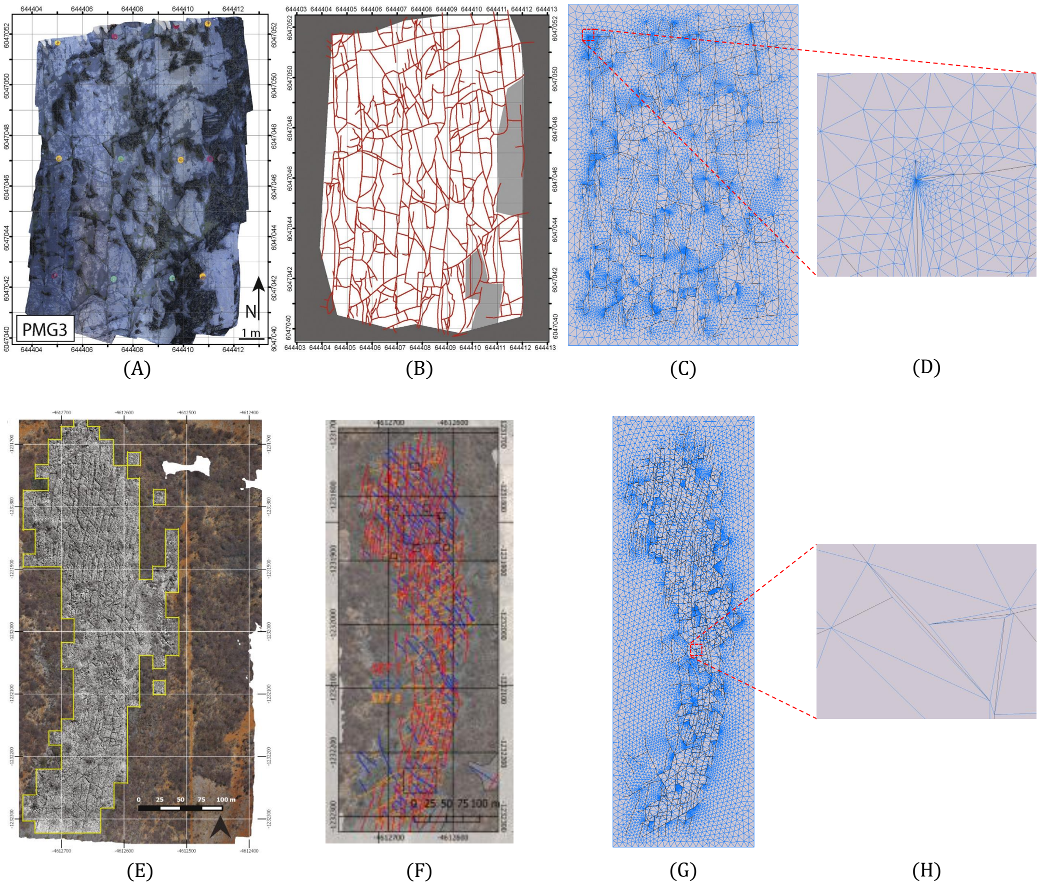


Figure 1. Fracture data acquisition, interpretation, and modeling steps. (A) and (E) Outcrop images obtained from the Whitby and Brejoes fieldwork area. (B) and (F) Manual interpretation of the fracture networks. (C) and (G) Conformal meshing results based on the raw interpretation. (D) and (H) is a zoom of the meshing artifacts due to complex fracture interaction. (A) and (B) Taken from Houben et al. (2017). (E) and (F) Taken from Q. Boersma et al. (2019).

al., 2001). Using the developed framework, we can create a fully conformal uniformly distributed grid based on any realistic fracture network at the required level of accuracy.

Most data obtained from outcrop studies is in planar 2D view (Bisdom et al., 2017). The available 3D data on fractures in the subsurface often consists of very coarse seismic cubes or borehole imaging logs. The attributes of the seismic cube are often too coarse to extract the exact fracture pattern, and the imaging logs only provide limited information at the well location (Q. Boersma et al., 2020). Therefore, this paper focuses on 2D fracture characterization and the preprocessing technique, which improves the meshing and subsequent fluid-flow modeling. We analyze the static and dynamic performance of the preprocessing on changes in geometry and topology of the fracture network and resulting mesh and changes in flow response. Ultimately, this leads to a robust way of constructing a hierarchy of DFMs for uncertainty quantification of natural fracture networks (de Hoop & Voskov, 2021).

Notice that the main ingredients of the developed framework and flow modeling are not limited to 2D and can be effectively applied for ~~a~~ fully 3D fracture~~d~~ networks (as shown by Karimi-Fard and Durlofsky (2016) from which we borrow several concepts). In 3D, all the fractures are represented by planes and discretized into segments (i.e., sub-planes) using an unstructured mesh. The vertices and edges of the meshed fractures will constitute the graph of the 3D fracture network, and the same preprocessing algorithm we propose in this paper (i.e., merging nodes) can be applied. Fracture apertures can be assigned to each edge of the discretized fracture, implying that the fracture aperture correction could also be used. However, some difficulties (e.g., projection of fracture aperture from the sub-plane to the edge and back) may introduce specific difficulties. A more straightforward approach could be making several slices through the 3D volume, projecting the fractures onto each slice and performing the same preprocessing on each slice (Sanderson et al., 2019).

The paper is organized as follows. We start with the description of the input data used in this study followed ~~up~~ by the theory for preprocessing, topology analysis, and fluid flow and energy transport modeling. Next, we describe all essential ingredients of the proposed framework, including intersection, node merging, straightening, and removing acute angles. The results section contains the analysis of the static and dynamic performance of the preprocessing framework. We finish the paper with a detailed discussion and conclusion.

2 Materials and Methods

The accurate numerical representation of fracture networks in the subsurface is not the end goal of the modeling effort. The modeling objective is often to make better predictions on subsurface activities and their associated risks. Therefore, it is essential to test our preprocessing framework accordingly. This is done by investigating the ~~static~~ changes introduced by the algorithm on the dynamic behavior of the subsurface (i.e., fluid flow response). Mainly, geothermal energy production is chosen (i.e., injection of cold water and production of hot water via a well doublet) to examine this. The methodology is presented here. First, a brief description of the fracture networks used in this work is given; second is a brief introduction to graph theory; third, a brief theoretical background on the topology of fracture networks is presented; fourth, the preprocessing method is presented; fifth, the relevant equations to model the physical processes are given; and, finally, the numerical approximation of governing equations is introduced.

2.1 Fracture network input data

The performance of the preprocessing algorithm is examined for two realistic fracture networks, a synthetic test case, and a variable aperture distribution applied to one of the realistic fracture networks. The first ~~fracture network~~ is found in the Whitby Mudstone outcrop along the cliff coast North of Whitby (UK) (Houben et al., 2017). The second example is the fracture network observed in the carbonate outcrop in Brejões, Brazil (Q. Boersma et al., 2019). Both networks are interpreted by hand; however, the developed method would also be very suitable for automatic fracture detection algorithms as presented in (Prabhakaran et al., 2019). The synthetic test case consists of a high permeable matrix and low permeable fractures with a narrow opening in the middle of the domain. The variable aperture model is applied to the Whitby fracture network (see Figure 2).

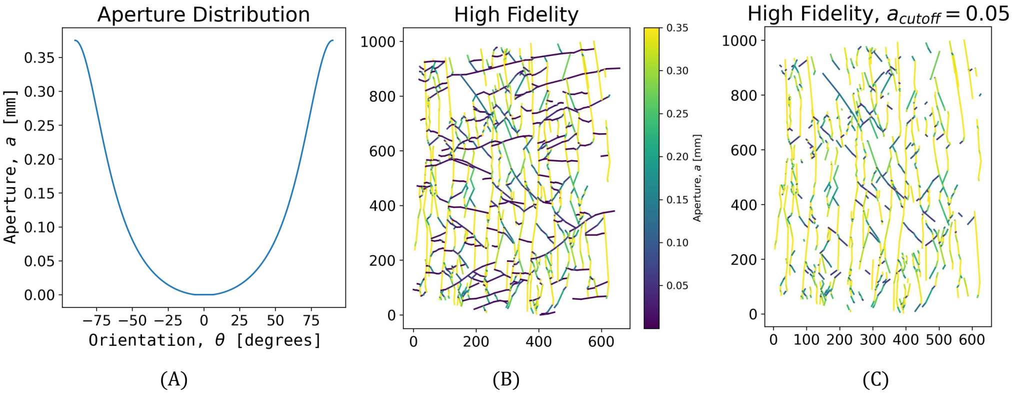
The outcrop images and the manual interpretation of the fracture networks are displayed in Figure 1. Both networks show good connectivity at first glance. The main difference between the two networks is the angle at which the fractures intersect. In the Brejoes network, this angle is around 60, while the angle is closer to 90 degrees for the Whitby network. The proposed fracture networks significantly differ in scale (Brejoes 100-1000 m vs. Whitby 1-10 m scale). Both networks are scaled up to characteristic reservoir size in a geothermal doublet system (Willems & Nick, 2019) by a scalar multiplication to preserve relative lengths and angles, which simplifies the static and dynamic analysis. The scalar is chosen for each network such that the resulting length in the y-direction is roughly 1000 [m] for both cases which is a typical distance between wells in a geothermal doublet system. ~~Conveniently, this is a common choice for the distance between two wells in a geothermal doublet system (Willems & Nick, 2019)~~. This scaling with a scalar multiplier can be safely done because of the fractal nature of fracture networks (i.e., the same pattern exists at several length scales) as discussed in Acuna and Yortsos (1995).

For the dynamic analysis, it is assumed that the two realistic fracture network models have very low permeability (i.e., convective flow is mainly limited by the fracture network) to ensure that the effect of changes to the fracture network on the flow response can be observed. ~~Most fracture network data is represented through the use of shapefiles. Shapefiles are typically transformed into a data array of size $m \times 6$, where the first column represents the id of the primary fracture, the second column represents the id of the subsegment, the remaining four columns represent the x- and y-coordinates of the two nodes associated with the subsegment, and m corresponds to the total number of fracture subsegments.~~ An important note is that not all manual interpretations record the intersection between all fracture segments (i.e., only the end nodes of the fractures are registered). This becomes important in the following section, where the graph is constructed based on the fracture network data. The two data arrays describing the fracture networks used in this study can be found by the following link: `https://github.com/MakeLikePaperrr/Fracture-Preprocessing-Code`.

To incorporate geological realism, a variable aperture distribution is applied to the Whitby fracture network, similar to the aperture model in Q. D. Boersma et al. (2021). The distribution and resulting apertures are visualized in Figure 2. Fractures oriented N-S are highly permeable, while fractures oriented E-W are low permeable. Figure 2 also depicts how a variable aperture leads to a much lower connectivity fracture network. Choosing a cutoff around 13% of the maximum conductivity leads to a large number of isolated fractures, hence, low connectivity.

2.2 Graph theory

As defined in Bollobás (2013), a graph G is an ordered pair of disjoint sets (V, E). The set of all vertices of graph G is denoted as $V = V(G)$, while the set of all edges of the graph G is denoted as $E = E(G)$. Edges of a graph join two vertices i and j such that $(i, j) \in E(G)$ and $i, j \in V(G)$. If $(i, j) \in E(G)$, it implies that i and j are *adjacent* vertices of G, and i and j are *incident* with the edge (i, j) .



(A) (B) (C)

Figure 2. (A) and (B) Aperture distribution as a function of angle (similar to Q. D. Boersma et al. (2021)). (C) Shows only the high-permeable fractures and illustrates that variable aperture models can lead to low connectivity fracture networks.

Important matrix representations of the graph G are the following four matrices:

1. Incidence matrix: $B(G)$, which is a $n \times m$ matrix, where n is the number of vertices and m the number of edges of the graph. As previously indicated, whenever a vertex i is on an edge $(i, \cdot)$, the vertex i is in incident with edge $(i, \cdot)$. Hence $B_{ij} = 1$ if vertex i is on the j-th edge otherwise $B_{ij} = 0$;
2. Degree matrix: $D(G)$, which is a $n \times n$ matrix describing the number of edges attached to each vertex. The degree matrix can be obtained using the follow equation $D = \mathtt{diag}\big(B\mathbf{1}\big)$, where $\mathtt{diag}(\mathbf{v})$ is a function that constructs a square matrix with vector $\mathbf{v}$ on its diagonal, and $\mathbf{1}$ is a vector of ones with size $m \times 1$. The degree matrix denotes the number of edges leaving a specific vertex.
3. Adjacency matrix: $A(G)$, which is a square $n \times n$ matrix, where n is the number of vertices of the graph G. As previously mentioned, if the pair of vertices $(i, j) \in E(G)$, they are said to be adjacent. Hence $A_{ij} = 1$ if vertices i and j are on the edge (i, j). Furthermore, for our purposes, it is assumed that the main diagonal is zero (i.e., $A_{ii} = 0$), which implies that no nodes are connected to itself. Note that A, B, and D are related through the following equation $A = BB^T - D$.
4. Discrete Laplacian matrix: $L(G)$ which can be found via the following equation $L = D - A = 2D - BB^T$. This matrix will be used for an alternative connectivity measure in the static analysis. The Discrete Laplacian is a matrix representation of the relationships defined in a graph.

A typical input data array $\mathcal{F}$ that describes the fracture network contains the pairwise x- and y-coordinates of each fracture segment in the network. The first step is to convert this array into two different forms: an array that contains all the unique vertices in the graph (i.e., V) and the incidence matrix (B). This is done by using Algorithm 1 which is found in the Appendix. An important assumption of this construction of V and B is that no subsegments can intersect in other places than the vertices of the particular subsegments. As mentioned before, this is often not the case in the manual interpretation of fracture networks; hence we need to calculate all possible intersections before applying Algorithm 1 shown in the Appendix. A simple intersection calculation algorithm is provided in Section 2.4.1.

2.3 The topology and geometry of fracture networks

In this section, the required mathematical relations for performing the static analysis on the effect of the preprocessing method on fracture networks are explained. Topology is used to understand how the connectivity and abutment-intersection relations of the fracture network are changing due to the preprocessing. Furthermore, it is also essential to look at how several geometrical properties of the fracture network are changing (e.g., angles and lengths) through preprocessing.

Several authors have thoroughly investigated the application of topology to fracture networks (Manzocchi, 2002; Sanderson & Nixon, 2015). Isolated nodes are typically denoted with an I, abutments are characterized by a Y-node, and X-nodes are used to indicate intersecting fracture segments. This is illustrated in Figure 3. Translating the type of nodes to the graph notation, node I is of degree one, node Y is of degree three, and node X is degree four. In general, it is unusual that more than two lines intersect at exactly one point. However, our preprocessing method merges nodes and causes several nodes to have a degree > 4. This causes us to consider all intersections of node degrees larger than four to be X type of nodes. This is used to plot the results in a ternary diagram (as shown in Figure 3). Classifying the fracture networks topology in this way allows us to use a proxy for the connectivity. Connectivity is often defined in this context as the average number of intersections per line. This changes slightly when allowing for nodes with a degree higher than four. Instead of the definition used in Balberg and Binenbaum (1983)

$$C_L = 4\frac{N_Y + N_X}{N_I + N_Y}, \tag{1}$$

where N_I is the total number of I nodes, N_Y is the total number of Y nodes, and N_X is the total number of X nodes, we use

$$C_L = 2\frac{\sum_i^d w_i N_i}{N_I + N_Y}, \tag{2}$$

where w_i are the weights, N_i the total number of i node types in the network, and $i = \{Y, X, X+, X++\ldots\}$. $X+$ represents a vertex which is one degree higher than an X (i.e., degree five instead of four), $X++$ two degrees higher, etc. The weights are determined by the number of lines (but not edges) involved in the vertex type (e.g., N_Y involves two lines therefore $w_Y = 2$ while N_{X++} involves six edges and hence three lines therefore $w_{X++} = 3$). Please also note that all vertices of degree two are not used nor important in this analysis (i.e., a curved or a straight line are topologically the same).

An alternative connectivity measure is obtained by using the Discrete Laplacian of the graph. This matrix can be used for finding spanning trees of a given graph (i.e., connected fracture sets in the fracture network). Notably, each element of the Laplacian's null-space rational basis describes a connected component of the graph (Spielman, 2010). With this basis, we can find the number of connected fracture sets in our network and also each fracture that belongs to these components (i.e., sub-graphs). The connectivity measure is then calculated as the ratio between the cumulative length of the fractures in the largest spanning cluster and the cumulative length of all the fractures in the full network.

Geometrical properties such as angles and lengths of the fractures are obtained using simple trigonometry rules. An easy and fast way to calculate the angles of a fracture w.r.t. the x-axis is to decompose the fracture into two components (i.e., $\Delta x = x_2 - x_1$ and $\Delta y = y_2 - y_1$). Then, the angle can be obtained using the following equation

$$\theta = \arctan\left(\frac{\Delta y}{\Delta x + \epsilon}\right), \tag{3}$$

where ϵ is a small perturbation to prevent the case of $\Delta x = 0$.

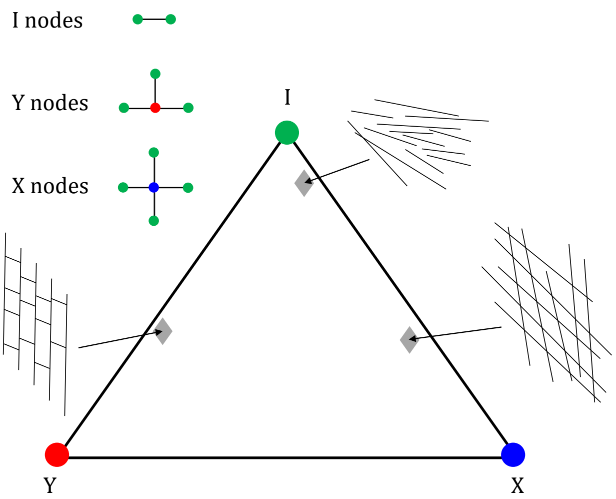


Figure 3. Illustration of topology in fracture networks. After Sanderson and Nixon (2015).

2.4 Preprocessing algorithm

For an accurate and efficient graph-based approach, a correct graph representation of the fracture network is necessary. Since not all intersections are always given via the fracture network's geological (or automatic) interpretation, we need to calculate all the intersections to construct the correct graph for a fracture network. After finding all the intersections, the large fractures are partitioned into smaller fracture segments with length l_f. Then, any two nodes that are too close in proximity are merged. Subsequently, segments that intersect at an angle below a certain threshold denoted as $\theta_{a,min}$ are merged as well. Furthermore, an optional straightening of the fractures can be applied to simplify the meshing procedure further if fractures intersect within $[180-\theta_{s,min}, 180+\theta_{s,min}]$. These steps are illustrated in Figure 4 and thoroughly explained in the following sections. Lastly, an aperture correction procedure is proposed, which deals with variable apertures and connecting previously disconnected fractures.

2.4.1 Intersections

Here the intersection detection method is described. The intersections are found by checking all combinations of any two edges. The combinations can be found via the binomial formula. All edges are parameterized, and a 2×2 linear system is solved for each pair of edges. Any intersection that occurs splits the two edges into four, and a vertex is added.

Let $\mathcal{X} = V \in \mathbb{R}^{n \times d}$ be the set of coordinates in the physical space of all unique vertices in the graph, where n is the number of vertices and d is the dimension of the physical space associated with the graph (i.e., fracture network). Then, let $\mathcal{P} = E \in \mathbb{R}^{m \times 2}$ be the set of all edges in the graph, where m is the number of edges and 2 represents the number of vertices associated with each edge. In other words, the j-th element of $\mathcal{P}$, $p_j \in \mathbb{N}^{2 \times 1}$, represents the set of two natural numbers associated with the two vertices of edge j. This means that $\mathcal{X}(p_j^1) = V(p_j^1, \cdot) = \mathbf{x}_j^1$ and $\mathcal{X}(p_j^2) = V(p_j^2, \cdot) = \mathbf{x}_j^2$, where $\mathbf{x}_j^1, \mathbf{x}_j^2 \in \mathbb{R}^d$ are the two vertices associated with edge j.

Finding all the intersections between any two edges, without any assumption on the location or orientation of the edge, can be done as follows. First parameterize all seg-

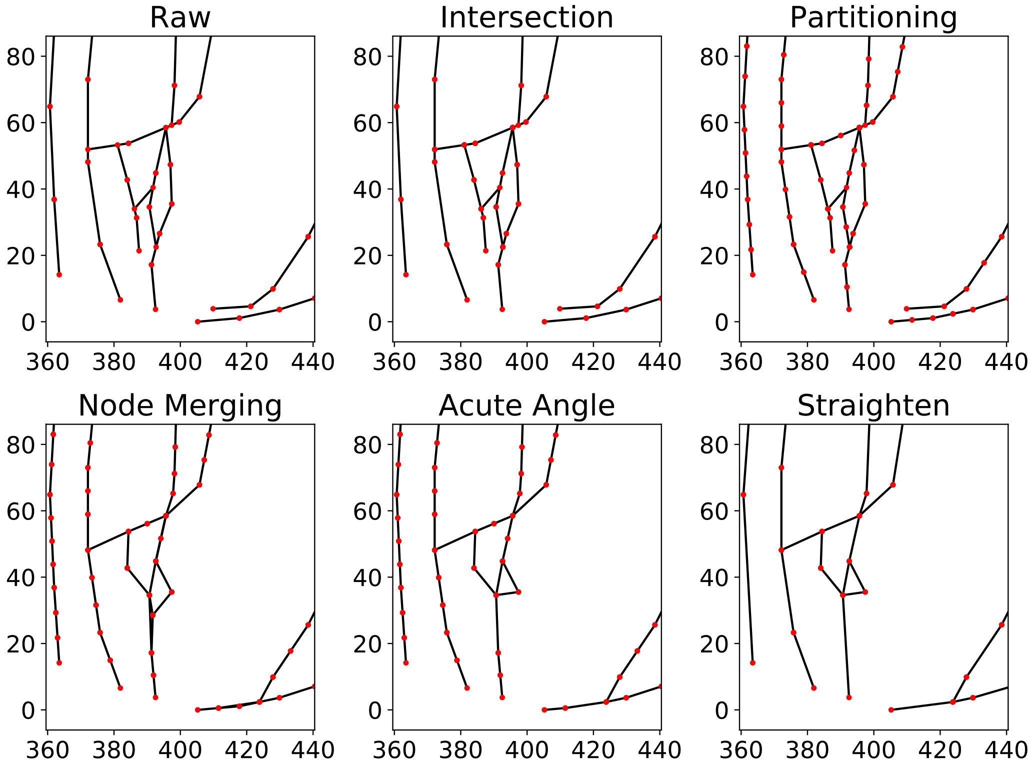


Figure 4. Illustration of the steps in the preprocessing workflow, from the raw data to a fully processed fracture network. The partitioning and node merging steps are a function of l_f while the acute angle and straighten steps are a function of $\theta_{a,min}$ and $\theta_{s,min}$ respectively. The smaller the l_f, the more precise the preprocessed network represents the raw data. However, small l_f means that the subsequent steps in the algorithm take substantially more time.

ments, using the following equation:

$$\mathbf{r}_j(t) = \mathbf{x}_j^1 + t\big(\mathbf{x}_j^2 - \mathbf{x}_j^1\big), \qquad j = 1, \ldots, m, \tag{4}$$

where $\mathbf{r}_j(t)$ represents a point on the fracture segments and t varies from 0 to 1 (from both end-points of the fracture segment). Find the pairs/combinations of edges, (i, j), that can possibly intersect,

$$\binom{m}{2} = \frac{(m)^2}{2}, \tag{5}$$

and solve the following equation for each such combination

$$\mathbf{r}_j(t) = \mathbf{r}_i(s). \tag{6}$$

The two edges intersect directly whenever $0 < t, s < 1$ is true (note: $<$ instead of $\leq$ indicates that the intersections at the end-points of segments are excluded). This simplifies to solving a $2 \times d$ system of equations for each intersection, such as

$$A\mathbf{x} = \mathbf{b}, \tag{7}$$

where $A = [\mathbf{x}_i^2 - \mathbf{x}_i^1, -\big(\mathbf{x}_j^2 - \mathbf{x}_j^1\big)]$, $\mathbf{x} = [t, s]^T$, and $\mathbf{b} = [\mathbf{x}_i^1 - \mathbf{x}_j^1]$.

The actual point of intersection is calculated by plugging the t that is obtained from Equation 7 into Equation 4. Every intersection involves exactly two segments, and the intersection id for those segments and x- and y-coordinate are stored in an array. After all the segments have been checked, a loop over this array allows us to manipulate intersections accordingly. For $\mathcal{X}$, this amounts to n_{int} new points, where n_{int} refers the the total number of intersection points. And for $\mathcal{P}$, each $p_j \in \mathcal{P}$ that contains at least one intersection gets replaced by $n_{int}^j{+}1$ new segments, where n_{int}^j refers to the number of intersections on the j-th segment.

This naive way of finding the intersection has the downside of having a large computational complexity (as indicated above). To circumvent this, we applied a method that takes advantage of the fact that most time is spent solving the linear 2×2 system in Equation 7. A simple check is applied for each pair of fracture segments to indicate if there can exist an intersection or not. Assuming the vertices of each edge (i.e., fracture) are ordered from smallest x-coordinate to largest, two edges can only have a possible intersection if the smallest x-coordinate of one of the two edges is smaller than the largest x-coordinate of the other edge (and vice versa for the y-coordinate). This significantly reduces the overall computational time of the algorithm as shown in the results section. Further reduction in computational time is achieved by parallelizing the algorithm, which is our ongoing development.

2.4.2 Node merging

The node merging algorithm, in essence, is sequential. Each vertex (i.e., node) is added to the domain ~~that~~ if it doesn't violate the algebraic constraint. This means that the distance between the newly added node and any other node already in the domain must be larger than $l_f \cdot h$, the node is merged into the closest node already in the domain. Parameter l_f refers to the accuracy at which the original fracture network will be processed and subsequently influences the optimal grid resolution, while h is a scaling parameter on the closed interval $[0.5, 0.86]$. The larger h is, the more simplified the resulting network becomes. Here, 0.5 is chosen as a lower bound such that any point on a fracture segment will get merged into one of the endpoints, while 0.86 is chosen as an upper bound such that any vertex perpendicular to the fracture segment with a distance equal to the height of an equilateral triangle with length l_f at the midpoint will get merged. The sequential nature of the algorithm implies that the order in which we add nodes to the domain affects the final result. Nodes that are added first are most likely placed in their exact location. Another essential consideration is the fracture aperture, since the conductive fractures often have a large impact on the fluid flow. Therefore, the fracture segments are ordered based on their aperture. If a single aperture model is applied, then segments are sorted based on the length of the fractures to minimize changes to the global structure of the network. ~~The larger the segment, the more critical it is for fluid flow, hence the earlier it should be added to the domain.~~

The length of each fracture segment, $L \in \mathbb{R}^m$, can be calculated in the following way:

$$L = \begin{pmatrix} ||\mathbf{x}_1^1 - \mathbf{x}_1^2|| \\ \vdots \\ ||\mathbf{x}_m^1 - \mathbf{x}_m^2|| \end{pmatrix}. \tag{8}$$

Then we define the order of adding segments, O_{segm}, from largest to smallest:

$$O_{\text{segm}} = \begin{cases} \{i \in \mathbb{N} \quad | \quad \forall l_i \in L, \quad l_i \geq l_{i+1}\}, & \text{if } \mathrm{a}_i = a \\ \{i \in \mathbb{N} \quad | \quad \forall a_i \in A, \quad a_i \geq a_{i+1}\}, & \text{otherwise} \end{cases} \tag{9}$$

where a_i is the aperture of fracture segment i, and A is the list of all fracture apertures.

From now on, for simplicity, it is assumed that $h = 1/2$. This means that $\frac{l_f}{2}$ is the minimum distance between each vertex in the simplified graph. To achieve this, a partitioning algorithm that divides each fracture segment in $m_i = \max(1, \text{round}(l_i/l_f))$ subsegments is executed. See Algorithm 2 for the detailed description.

Now we can construct the graph representation of the ordered and partitioned fracture network, using Algorithm 1 and substituting $\mathcal{F}$ with $\mathcal{F}_{\text{new}}$ and m with m_{new}. Furthermore, the problem is that vertices are added to the domain and not necessarily edges. Therefore, we need to determine the order in which vertices should be added to the domain. The order of the vertices, O_{vertices}, can be found with Algorithm 3.

After the order is determined and B and $\mathcal{X}$ are sorted, the primary node merging algorithm can be applied. It simply consists of sequentially checking, from highest to lowest priority vertices, if a newly added node violates the algebraic constraint (i.e., is within $\frac{l_f}{2}$ from any nodes already in the domain). This is thoroughly described in Algorithm 4.

The main parameter in the partitioning and subsequent node merging algorithm is the preprocessing accuracy l_f. This parameter determines the minimum distance between any vertex in the simplified graph. The computational time of the algorithm scales proportionally to the l_f and the number of fractures.

2.4.3 Straightening and removing acute angles

Another (optional) modification to the fracture network is the straightening of fracture segments. This amounts to checking each vertex with order two and calculating the angle between the two edges leaving this vertex. If this angle is within some threshold, particularly within $[180 - \theta_{s,min}, 180 + \theta_{s,min}]$, the node can be removed since the fracture is considered straight. The angle $\theta_{s,min}$ is typically chosen on the interval [0, 7.5], depending on how severely the user wants to straighten the fractures. The straightening of fractures can be beneficial when considering meshing tools such as GMSH (Geuzaine & Remacle, 2009). The reason for this is that conformal meshing techniques require the fracture to be embedded into the domain. Less embedded fractures mean faster and easier meshing.

Simply merging the conflicting nodes doesn't resolve all the artifacts associated with meshing DFMs. This is mainly caused by the fact that the algebraic constraint, $\frac{l_f}{2}$, is constant. Whenever nodes are merged, the corresponding edge (i.e., fracture segment) might be stretched and have a length greater than l_f. This might result in vertices being placed near existing edges and not flagged as problematic nodes by the node merging algorithm. Therefore, ~~an need~~ an additional correction to the network is required to obtain the optimal representation for meshing purposes.

The algorithm for removing the acute angles is very similar to Algorthm 5; however, now the loop is over all nodes with a degree bigger than one. Instead of calculating one angle, $\binom{d_i}{2}$ angles are computed between all edges leaving the vertex i, where d_i is the degree of vertex i. The two edges corresponding to the smallest angle below a certain threshold will be merged. The smaller segment will be merged in the larger segment, and the non-coinciding vertex will be merged in the closest vertex of the larger segment. This ensures minimal changes to the fracture network due to other possible edges leaving the merged vertex. The tolerance for the minimal angle $\theta_{a,min}$ is typically chosen on the interval [0, 18] degrees. Larger $\theta_{a,min}$ means a more simplified fracture network since potentially more fracture intersections are flagged as problematic.

2.4.4 Aperture correction

In order to incorporate low connectivity and fracture networks with variable apertures, a fracture aperture correction is applied during the cleaning procedure. Connected fractures are treated analogous to resistors in an electric circuit. Resistance is equal to the inverse of the hydraulic conductivity, which in turn is a function of the square of the fracture aperture. Figure 5 displays the two different corrections (Type 1 vs. Type 2). Type 1 corrections result in the overlap of two segments after merging of vertices. Type 2 is subdivided further into 2a, which results in an edge collapse, and 2b, which connects two previously disconnected edges. The following equation gives the correction for Type 1

$$\hat{R} = \left(\sum_{i=1}^{n=2} \frac{1}{R_i} \right)^{-1}, \tag{10}$$

where R_i is the resistance of edge i defined as

$$R_i = \frac{L_i}{a_i^2}, \tag{11}$$

where L_i is the length and a_i is the aperture of the i-th edge respectively, such that the effective aperture of the corrected edge is given by

$$\hat{a} = \sqrt{\hat{L} \sum_{i=1}^{n=2} \frac{a_i^2}{L_i}}, \tag{12}$$

where $\hat{L}$ is the length of the new edge.

The Type 2a correction is given by

$$\hat{R} = \sum_{i=1}^{n=2} R_i, \tag{13}$$

resulting in

$$\hat{a} = \sqrt{\frac{\hat{L}}{\sum_{i=1}^{n=2} \frac{L_i}{a_i^2}}}. \tag{14}$$

In the case of Type 2b, an effective matrix aperture is obtained by inverting the permeability of parallel plate flow such that

$$a_{mat} = \sqrt{12 k_{mat}}, \tag{15}$$

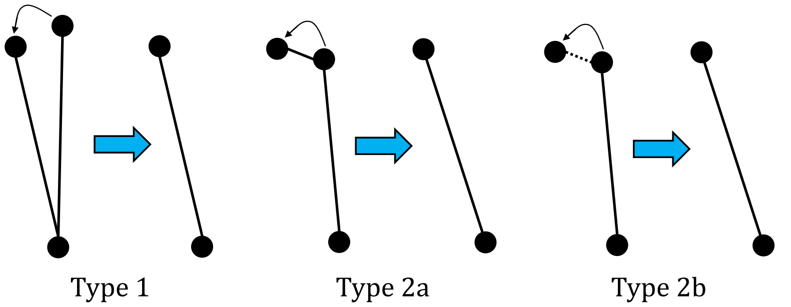


Figure 5. Illustration of the types of aperture corrections. Merging the non-shared vertex results in a Type 1 correction (parallel resistor), while merging the shared vertex results in a Type 2 correction (sequential resistor). An effective matrix aperture is used if the two edges are disconnected (Type 2b).

where k_{mat} is the matrix permeability at the location of the particular edge. A further addition to the Type 2b correction is added to preserve the characteristics of impermeable fractures/faults and high permeable matrix, given by

$$\hat{a} = \begin{cases} \frac{1}{\hat{L}}\Big(a_{mat}L_{mat} + a_i L_i\Big), & \text{if } \mathrm{a}_{mat} > a_i \\ \dfrac{1}{\frac{1-\left(L_i/(L_i+L_{mat})\right)^n}{a_i} + \frac{\left(L_{mat}/(L_i+L_{mat})\right)^n}{a_{mat}}}, & \text{if } \mathrm{a}_{mat} \leq a_i, \end{cases} \tag{16}$$

where L_{mat} is the gap between the vertices and n is determined by fitting a least-squares solution to tracer simulation on two disconnected fractures with different gaps and characteristic cleaning lengths, given by

$$n = 9.56\frac{L_{mat}}{L_i + L_{mat}} + 1.18. \tag{17}$$

If $n \approx 1$ we have the normal harmonic mean, while $n \to \infty$ is the same as not applying any aperture correction. The parameter n effectively limits the aperture penalty when connecting disconnected fractures, as it was observed from simple numerical experiments that the aperture correction, in some cases, over-penalizes the effective aperture. Also, note that n is bounded since L_{mat} can never exceed $l_f h$, since otherwise, these vertices would not apply for merging.

To deal with vertices that are connected through a path in the neighborhood of the vertices, a sub-graph is extracted around the vertex that is merged. The shortest path is computed using Dijkstra's algorithm implementation described in Csardi et al. (2006). Whenever there is no shortest path (i.e., even in the neighborhood the two vertices remain disconnected), the effective matrix aperture is used for the resistance instead. This is represented in Figure 6

Figure 6 illustrates an essential feature of the aperture correction. Merging vertex 12 into vertex 11 results in a reduction of the aperture of the edges connecting vertex 10 and 12 as well as 12 and 13. This is undesirable because we want to preserve this connectivity. Since the vertex merging happens sequentially, it has become evident at this point that sorting based on aperture (highest to lowest) is more effective for an accurate representation of fluid flow in the fracture network than simply sorting based on length.

All the code related to the algorithms described above is implemented in Python and can be found at `https://github.com/MakeLikePaperrr/Fracture-Preprocessing`

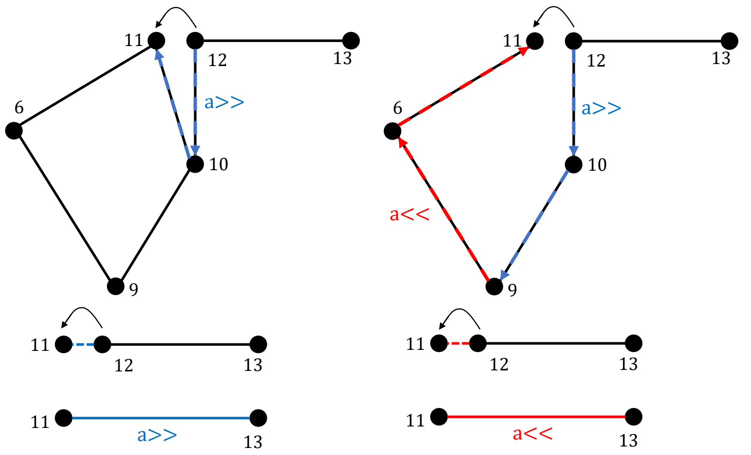


Figure 6. Vertices that don't share an edge might be connected through a path in the neighborhood of the vertices. A sub-graph is extracted and using Dijkstra's shortest path; the effective resistance is computed. If no shortest path exists (i.e., even in the neighborhood the two vertices remain disconnected), the effective matrix aperture is used instead.

`-Code`. We have made use of the following packages: NumPy (Harris et al., 2020), SciPy (Virtanen et al., 2020), and igraph (Csardi et al., 2006).

2.5 Governing equations

In order to evaluate the dynamic performance of the preprocessing algorithm, several flow scenarios are considered, for which the governing equations are specified here. The conservation of mass, in general form, is written as

$$\frac{\partial}{\partial t}\Big(\phi \sum_{p=1}^{n_p} x_{cp}\rho_p s_p\Big) + \nabla \cdot \sum_{p=1}^{n_p} x_{cp}\rho_p \mathbf{v}_p + \sum_{p=1}^{n_p} x_{cp}\rho_p q_p = 0, \quad c = 1,\dots,n_c \tag{18}$$

where ϕ represents the porosity, x_{cp} is the molar mass fraction of component c in phase p, ρ_p is the density, s_p is the saturation, and q_p is the source term of the p-th phase respectively, and $\mathbf{v}_p$ is the velocity of the p-th phase. The Darcy velocity of the p-th phase is given by

$$\mathbf{v}_p = -\frac{k_{r,p}}{\mu_p}\mathbf{K}\nabla(p_p - \rho_p \mathbf{g}), \qquad p \in \{o, w\} \tag{19}$$

where $k_{r,p}$ is the relative permeability, μ_p is the viscosity and p_p is the pressure of the p-th phase respectively, $\mathbf{K}$ is the permeability tensor, and $\mathbf{g}$ is the directional gravitational acceleration defined as $g\nabla z$. The equations for the tracer simulation, applied to the variable aperture model, are obtained by having a two-component single-phase system and setting the density and viscosity equal to unity.

The following equation describes the conservation of energy required for the geothermal simulations:

$$\frac{\partial}{\partial t}\left(\phi \sum_{p=1}^{n_p} \rho_p s_p U_p + (1-\phi)U_r\right) + \nabla \sum_{p=1}^{n_p} h_p \rho_p \mathbf{v}_p + \nabla(\kappa \nabla T) + \sum_{p=1}^{n_p} h_p \rho_p q_p = 0, \tag{20}$$

where U_p is the internal energy of fluid phase p, U_r is the rock internal energy, h_p is the enthalpy of phase p, κ is the thermal conduction, and T is the temperature. All governing assumptions and properties can be found in (Wang et al., 2020a, 2021).

2.6 Numerical solution

Finite-volume discretization is applied to a general unstructured grid (using a Two-Point Flux Approximation (TPFA) for the fluxes across interfaces with upstream weighting) and a backward (implicit) Euler time discretization strategy to both the conservation equations and obtain the following system of equations (assuming no gravity and capillarity)

$$\begin{aligned} V\left[\left(\phi\sum_{p=1}^{n_p} x_{cp}\rho_p s_p\right)^{n+1} - \left(\phi\sum_{p=1}^{n_p} x_{cp}\rho_p s_p\right)^{n}\right] - \Delta t\sum_l\left(\sum_{p=1}^{n_p} x^l_{cp}\rho^l_p\Gamma^l_p\Delta p^l\right)&, \\ + \Delta t\sum_{p=1}^{n_p}\rho_p x_{cp} q_p = 0, \quad c = 1,\ldots,n_c& \end{aligned} \tag{21}$$

and

$$\begin{aligned} V\left[\left(\phi\sum_{p=1}^{n_p}\rho_p s_p U_p + (1-\phi)U_r\right)^{n+1} - \left(\phi\sum_{p=1}^{n_p}\rho_p s_p U_p + (1-\phi)U_r\right)^{n}\right]& \\ - \Delta t\sum_l\left(\sum_{p=1}^{n_p} h^l_p\rho^l_p\Gamma^l_p\Delta p^l + \Gamma^l_c\Delta T^l\right) + \Delta t\sum_{p=1}^{n_p} h_p\rho_p q_p = 0,& \end{aligned} \tag{22}$$

where Γ^l_p is the convective and Γ^l_c is the thermal transmissibility of interface l and phase p respectively.

For details regarding the handling of the fractures, the reader is referred to Karimi-Fard et al. (2004). In short, fractures constitute a control volume in the computational domain of a similar dimension as the matrix control volumes, particularly approximated by a porous media where the permeability follows from the parallel plate model (cubic law, i.e., $k_f = a^2/12$). This allows solving the above system of nonlinear equations everywhere in the domain without using fracture-matrix transfer functions. The mass and heat transfer between fracture and matrix naturally follows from the discretization.

Operator-Based Linearization (OBL) is used to linearize the above system of nonlinear equations (i.e., Equation 21 and 22). OBL is a novel way of performing the linearization step. The discrete form of the mathematical equations is grouped into state-dependent operators and space-depended relations. The parameter space of the problem is discretized, where each axis is split by the uniformly distributed set of supporting points. Any point in the parameter space belongs to a certain hypercube bounded by supporting points. Next, the nonlinear operators are subsequently calculated exactly in a set of supporting points at a preprocessing stage or adaptively. At the simulation stage, the operators' values and their derivatives are evaluated using multi-linear interpolation inside a particular hyper-cube in the parameter space where the specific simulation state belongs. The multi-linear interpolation of the most nonlinear part of the governing equations provides simple, exact, and above all flexible Jacobian assemble for the nonlinear solution procedure. For details on the OBL framework, the reader is referred to Voskov (2017); Khait and Voskov (2017, 2018a).

The proposed fracture network processing framework has been fully integrated with the open-source Delft Advanced Research Terra Simulator (DARTS). DARTS is a scalable parallel simulation framework, which has been successfully applied for modeling of energy transition applications, including hydrocarbon (Khait & Voskov, 2018b; Lyu, Khait, & Voskov, 2021), geothermal (Khait & Voskov, 2018c; Wang et al., 2020b) and CO_2 sequestration (Kala & Voskov, 2020; Lyu, Voskov, & Rossen, 2021) cases. The ongoing effort to include fully coupled geomechanical modeling into DARTS allowed us to directly address induced seismicity problems (Novikov et al., 2021) which is another challenge in energy transition usually directly relevant to fracture networks.

3 Results

This section presents the investigation of the performance of the preprocessing method described in the previous section. The performance is assessed in terms of static and dynamic qualities and is therefore subdivided accordingly. It is important to stress the difference between the preprocessing accuracy l_f and the meshing accuracy l_m. The parameter l_f refers to the minimum distance between any two vertices in the preprocessed fracture network. In contrast, l_m refers to the characteristic length of the control volumes after applying a particular meshing strategy (i.e., Frontal-Delaunay as a 2D meshing algorithm in this work, see Geuzaine and Remacle (2009) for details).

Following the definition of those two parameters, there is a significant distinction between the two preprocessing strategies described below. The first approach is defined as the “clean” strategy. In this approach, the preprocessing algorithm is executed once with a $l_f = 1$, $\theta_{a,min} = 18°$, and $\theta_{s,min} = 2.5°$. The l_f remains unchanged in the clean strategy for subsequent coarser meshing results. The second strategy is denoted as the “optimal” strategy. In this strategy, for each subsequent coarser model, the preprocessing algorithm is executed with $l_f = l_m$. This means that the fracture network in the “clean” strategy remains unchanged when coarsening the mesh. In the “optimal” strategy, the fracture network changes when constructing the coarser models.

3.1 Static performance of the preprocessing framework

3.1.1 Changes in configuration

Figure 7 illustrates several changes to the raw fracture network after applying successive coarsening. An apparent reduction in the number of nodes (red dots) can be seen with increasing l_f, which significantly reduces the number of fracture segments. Fewer fracture segments typically indicate a lower network complexity (simply by having fewer degrees of freedom). Multilinear segments become linear (i.e., straight) because of the reduction in fracture segments, further reducing network complexity. Ultimately, small and complex features of the fracture network start to disappear while the main pattern (backbone) remains visible. The average spacing of the North-South fractures (40-50 meters) remains unchanged up to $l_f = 32$. Around $l_f = 64$, which exceeds this average spacing, the fracture configuration changes substantially, as shown in Figure 8.

3.1.2 Angle distribution

A critical characteristic in fracture networks is the angle distribution, particularly weighted by the length of the fractures, especially when considering variable apertures (Baghbanan & Jing, 2008; Bisdom et al., 2016). This usually gives an insight into the potential flow response of the network while also providing possible information on the paleostress that caused the network formation. Since multiple nodes are merged in the preprocessing approach, it is expected that these angles can change substantially when using a large l_f, where large is relative to the scale at which the raw data is collected. This can be clearly seen when looking at Figure 8. For small l_f, the deviation in angles is almost unnoticeable, while around $l_f = 32$, a small deviation of roughly 10% in the orientation is observed in the Whitby network. Around $l_f = 64$, the deviation becomes significant ($> 20\%$), but the dominant orientation (N-S) is still similar to that of the raw results. Finally, at $l_f = 128$, the angle distribution is very different from the raw data ($> 30\%$), even the dominant orientation, and doesn’t resemble the original network.

Similar behavior but at earlier resolution is observed for the Brejoes network. At $l_m = 16$ the deviation is roughly 20%. The dominant NNW-SSE orientation disappears already at $l_f = 64$. The average spacing of the NNW-SSE fractures in the Brejoes network is roughly 12 meters. This shorter spacing correlates with the earlier deviation in the angle distribution in the Brejoes network when compared to Whitby.

3.1.3 Topology

Besides the angle distribution, it is also important to look at connectivity and in particular, the topology changes to the fracture network. Figure 9 shows the topology of the raw and preprocessed fracture networks in the ternary topology diagram (as explained in Figure 3). A large deviation between the raw and preprocessed data is observed, even with the small $l_f = 1$ [m]. The raw network contains roughly 55% I-nodes, 20% Y-nodes, and 25% X-nodes. The finest preprocessed network (i.e., $l_f = 1$ [m]) contains approximately 20% I-nodes, 75% Y-nodes, and 5% X-nodes. Furthermore, with increasing l_f, the preprocessed networks increasingly deviates towards a large X-node percentage (from 5% at $l_f = 1$ to almost 70% at $l_f = 128$ for Whitby and from 10% at $l_f = 2$ to 70% at $l_f = 64$ for Brejoes).

To illuminate the differences in topology between the fine $l_f = 1$ and the raw data, the degree of the raw and cleaned network nodes is shown in Figure 10. Even after zooming in at the nodes of the raw network, a significant amount remains misclassified as I-nodes while they would be more suitably classified as Y-nodes or X-nodes (at this scale of observation). This is the result of two fracture segments essentially intersecting, but not exactly due to inaccuracy in image interpretation. The same behavior arises for the

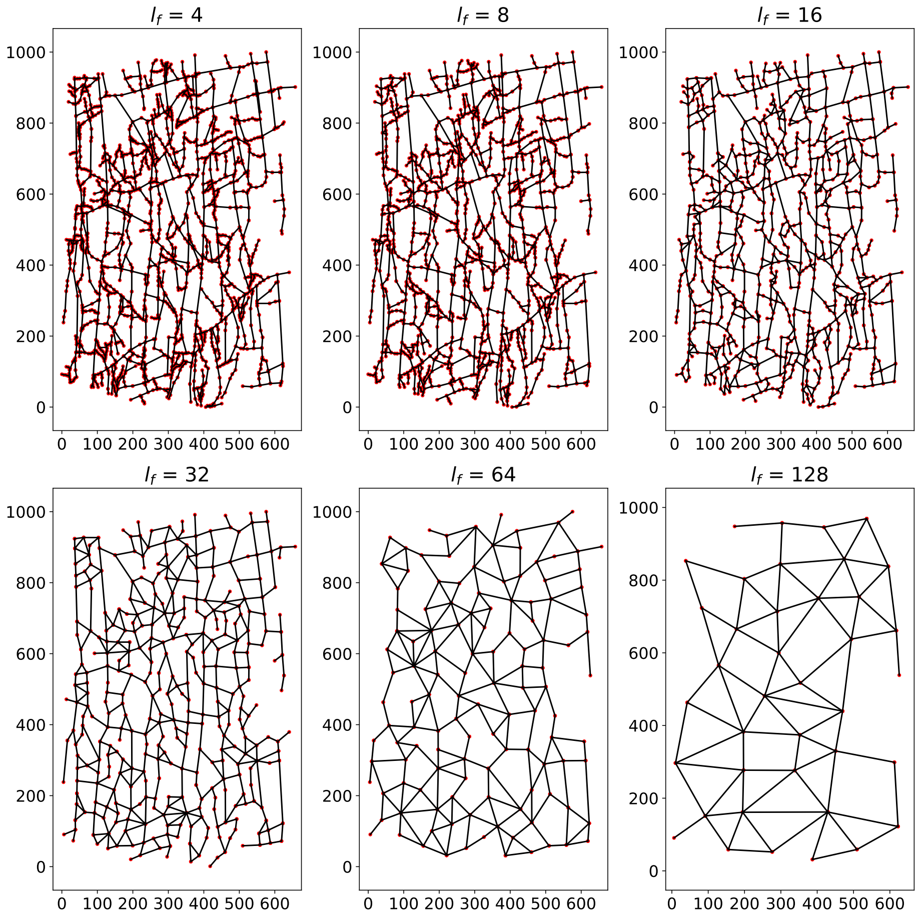


Figure 7. Changes to fracture network as a function of preprocessing accuracy l_f. The network's complexity is greatly reduced by the decrease in fracture segments with increasing l_f. The angles of the N-S fractures remain unchanged up to $l_f = 64$ [m].

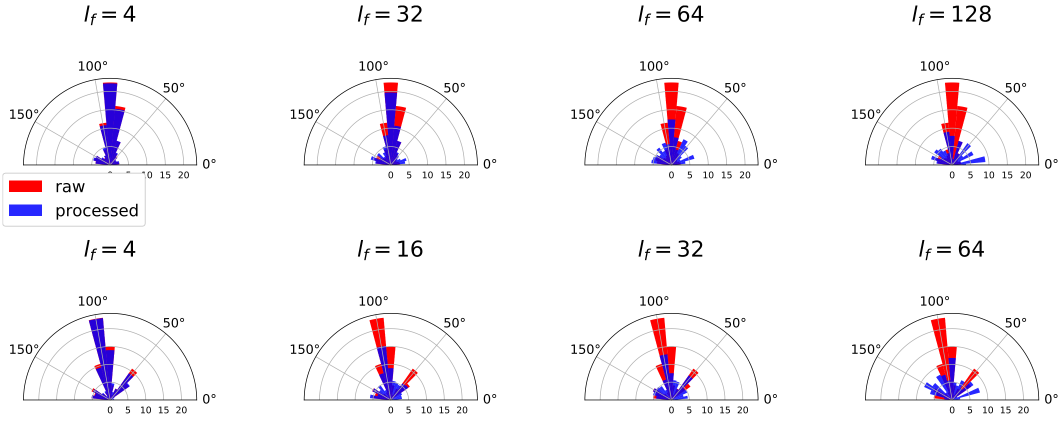


Figure 8. Angle distribution as a function of fracture cleaning accuracy. The top row corresponds to the Whitby network, while the bottom row corresponds to the Brejoes network. The cleaning shows no significant change between $l_f = 4$ and $l_f = 16$ for the Whitby network; that's why these steps are omitted in the figure. However, the Brejoes network does show significant deviation at $l_f = 16$. The preprocessed Whitby network is no longer representative of the raw network at $l_f = 128$, while this already happens at $l_f = 64$ for the Brejoes case.

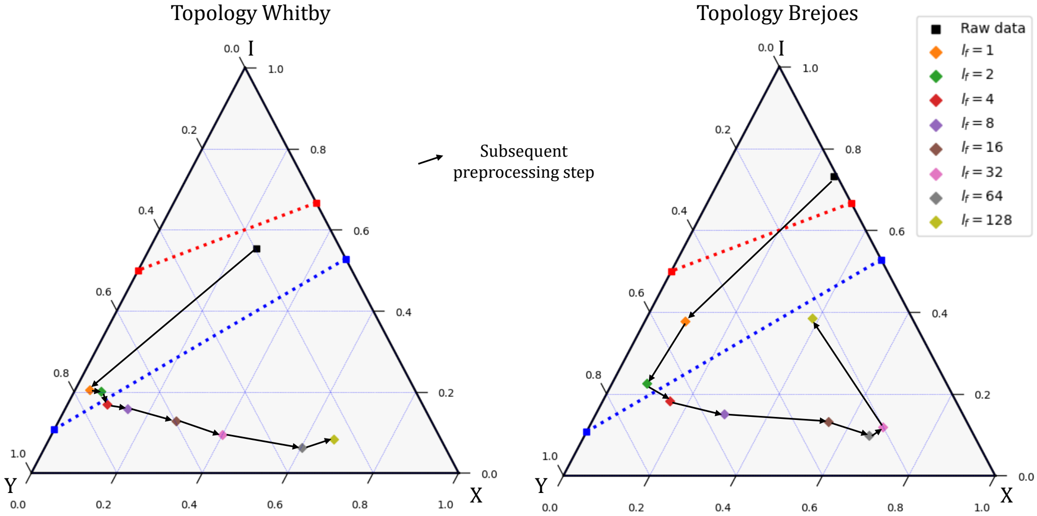


Figure 9. A large deviation between the raw data and the processed network's topology in both fracture networks is observed. The reason for this is explained in Figure 10. The Brejoes network converges to the raw data for low $l_f < 1$. The jump in the large $l_f = 128$ for the Brejoes case is expected due to the fracture network becoming extremely coarse. Only a few fractures actually remain, meaning the relative proportion of end-nodes greatly increases.

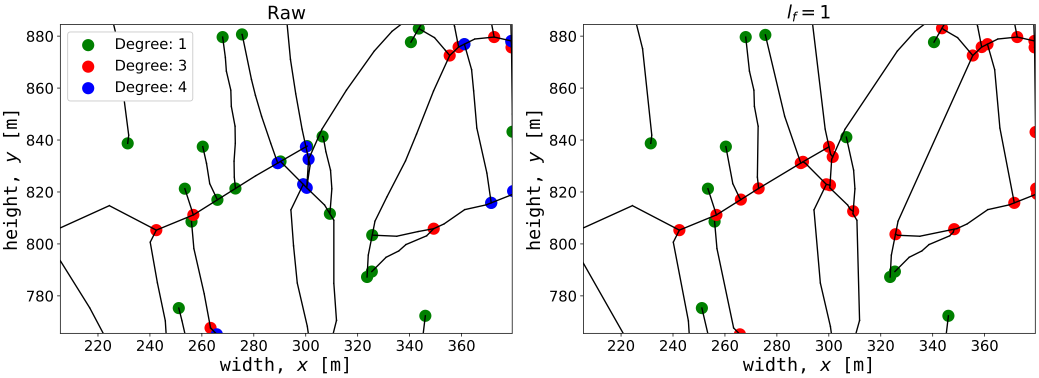


Figure 10. Detailed view of the fracture network topology of the Whitby network. The left image displays the raw input topology, while the right image shows the topology after applying the preprocessing algorithm with $l_f = 1$. Due to the manual interpretation, it can be seen that a lot of nodes are characterized as I-nodes (degree 1) or X-nodes (degree $\geq$4) in the left plot, while most seem to be Y-nodes (degree 3) (when considering usual abutment relationships in fracture mechanics and the resolution of the outcrop image).

X-nodes that are misclassified as Y-nodes. This happens when two fracture segments only intersect with a minimal extension of one of the segments across the point of intersection.

3.1.4 Impact of changes on meshing

Because the complexity of the fracture network decreases, the conformal meshing procedure becomes substantially easier. This is shown in Figure 11. A significant reduction in the number of control volumes and a more homogeneous distribution is observed for the preprocessed meshing results compared to the raw network. The dark blue areas in the raw meshing results indicate a concentration of small control volumes. Furthermore, very flat triangular elements are observed at some locations in the raw meshing results. Therefore, it seems that the volume distribution and quality of the mesh elements are improved in the preprocessed results. This is quantified in Figure ,12 and Figure 13 respectively. Please note that the fluid flow simulations are carried out in the 3D domain and therefore the model is assigned a thickness (2.5D).

Mesh quality here refers to a similar definition as used in Mustapha and Dimitrakopoulos (2011), particularly using the following equation

$$q = 4\sqrt{3}\frac{A}{a+b+c}, \tag{23}$$

where A is the area of the triangle and a/b/c are the lengths of the three sides of the triangle, respectively. This means that when $q = 1$ we have a high mesh quality since the triangle is equilateral (i.e., the optimal shape for TPFA fluid-flow simulation), while a low-quality mesh element (i.e., $q << 1$) refers to a large deviation from an equilateral triangle. The mesh elements in the 2.5D model are triangular prisms which means that this mesh element quality indicator also works for this type of geometry. The reason for this is that the centroid of the triangular prism lies in the same xy-plane as the centroid of the triangle and is therefore not changing the orthogonality relationship between neighboring control volumes.

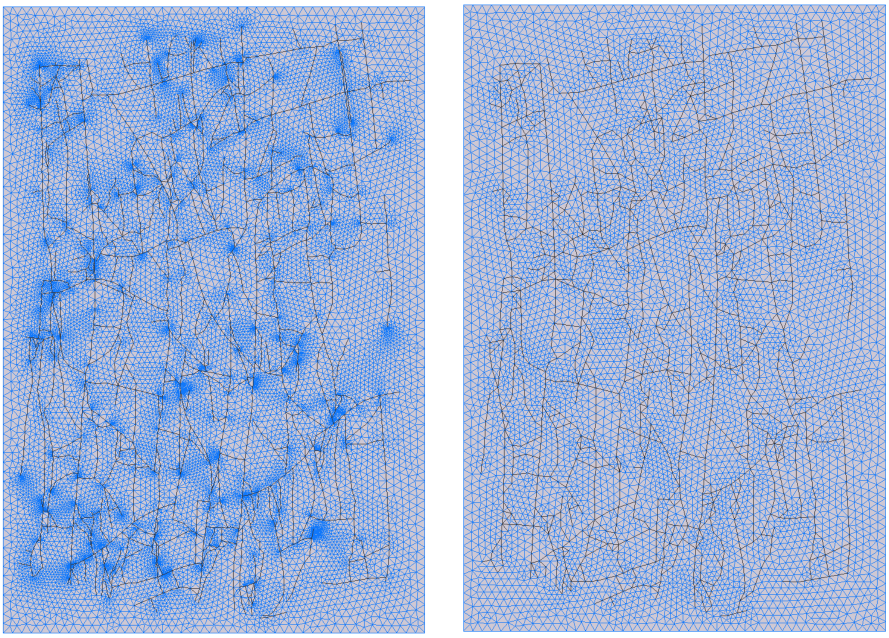


Figure 11. Visual comparison between the meshing result of the raw (left) versus the cleaned (right). Meshing and preprocessing accuracy are both 32 [m] (i.e., $l_m = l_f = 32$). The darker blue spots in the image on the left represent clusters of small control volumes. These appear at locations of complex fracture interactions on a scale way below the meshing resolution l_m.

Ultimately, the purpose of using and generating fracture networks is to utilize them in specific industrial applications. In this work, the chosen application is geothermal energy production from the subsurface. This application usually requires multiple numerical simulations to address general uncertainty in subsurface parameters. The accuracy and speed of convergence of these simulations are highly dependent on the mesh quality and, specifically for fracture networks, the orthogonality of the control volume intersections and the volume distribution. Therefore, we quantify the effect of the preprocessing method on these two properties, where mesh quality is a proxy for the orthogonality of the control volume intersections. Figure 12 shows the volume distribution as a function of l_f and l_m, while Figure 13 shows the distribution of mesh element quality.

The volume distribution obtained after meshing the raw fracture network input is not normally distributed. It has a peak around zero, which indicates a large number of small control volumes. This effect becomes more substantial with increasing l_m. At $l_m = 32$ the volume distribution of the raw network input is entirely concentrated around zero. The volume distribution obtained after meshing the optimal preprocessed fracture network input does show a normal distribution. The distribution becomes wider and more skewed with increasing the l_m. No small control volumes are observed for the optimal preprocessed results, even in $l_m = 128$ [m]. The clean preprocessing strategy shows similar behavior to the optimal strategy for small l_m, while converging to the behavior of the raw input network for $l_m \geq 32$.

The mesh element quality obtained after meshing with a small l_m behaves similarly for the raw and preprocessed input fracture data, except for a relatively small amount of flat triangles (i.e., $q \approx 0$). An increase in the number of flat triangles (i.e., $q \leq 0.01$) from 0.32% to 1.29% and a reduction of the overall quality is observed for the raw input data with increasing l_m. However, the mesh quality for the preprocessed results re-

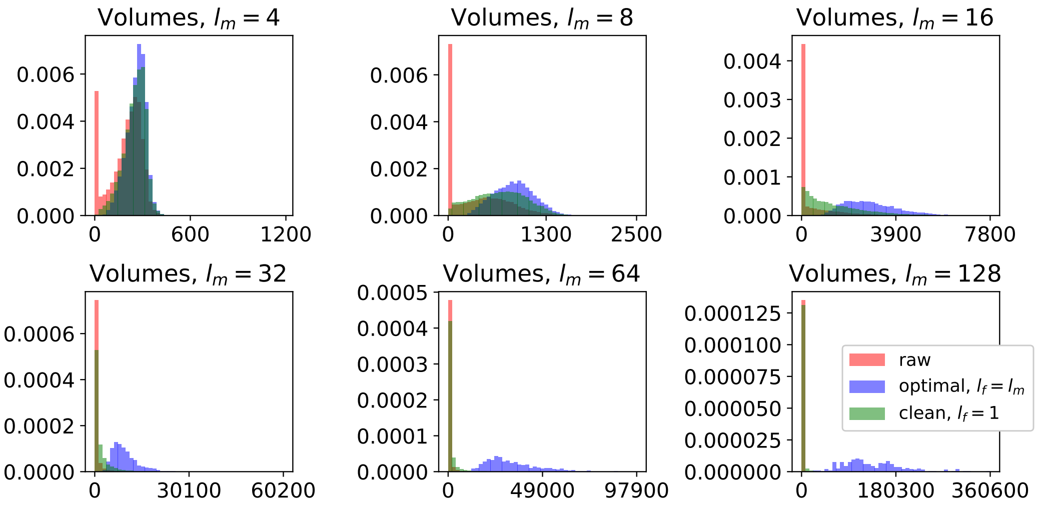


Figure 12. Control volume size distribution as a function of preprocessing accuracy for the Whitby network. Optimal refers to the preprocessing strategy where the fracture network is cleaned at the same accuracy as the mesh is generated. Clean refers to preprocessing the fracture network once at a small l_f and then simply decreasing the meshing resolution l_m wile keeping the fracture network unchanged.

mains above $q = 0.40$ even for $l_m = 128$ [m]. Low mesh quality (i.e., $q \leq 0.01$) can be seen as an indicator for poor simulation convergence since a few of these elements can ruin the nonlinear convergence behavior of the numerical simulation (more than the mean mesh element quality or the whole distribution).

3.2 Dynamic performance of preprocessing framework

3.2.1 High enthalpy single aperture

The dynamic performance is analyzed by applying geothermal simulation to the different DFM models obtained after meshing (i.e., clean and optimal for different l_m). Geothermal simulation typically consists of a doublet system: at one point, cold water is injected, and at another point, hot water or steam is produced. Mathematically speaking, this amounts to solving Equation 18 and 20 presented in Section 2.5. The injection point is in the bottom left of the domain, while the production point is at the top right of the domain. Both wells are perforating a fracture segment. First, the temperature fields of both networks are shown (Figure 14 and 15). The water saturation field is shown for the Brejoes network (Figure 16), and finally, the temperature at the production well over time (Figure 17).

The boundary conditions and modeling parameters can be found in Table 1 and 2. The simulation parameters model a situation that is investigated throughout the world for its geothermal energy potential (Moeck, 2014). Particularly, we study geothermal energy production from a tight fractured reservoir with convective flow happening predominantly through the fracture network. It is important to observe how changes to the fracture network affect the simulation results in such a setup. If the fracture permeability is much larger than the matrix permeability, the fractures will evidently play a dominant role in the fluid flow patterns. There are a particular set of parameters for each network. The first set of parameters simulates initially high-enthalpy single-phase supercritical water according to IAPWS 97 equation of state (Wagner & Kretzschmar, 2008)

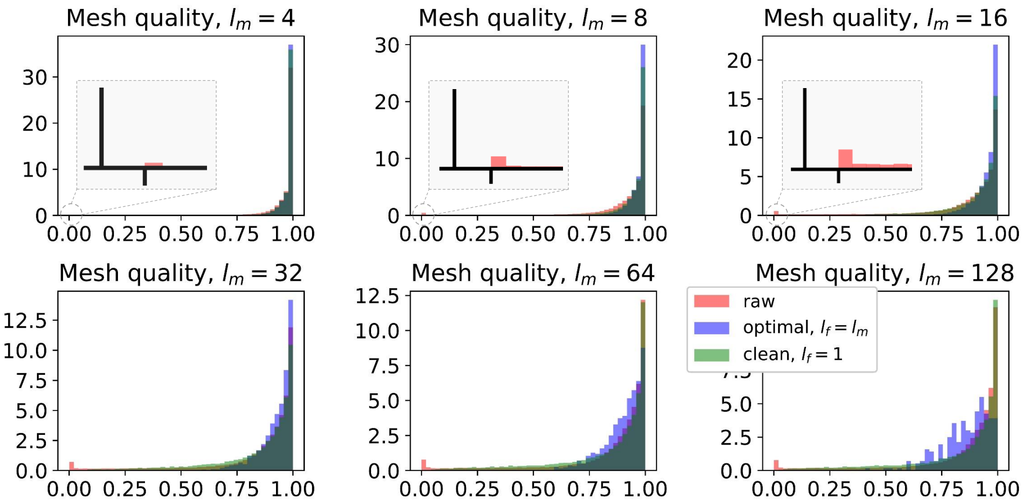


Figure 13. Mesh element quality distribution as a function of preprocessing accuracy for the Whitby network.

used in DARTS. This set of parameters is applied to the Whitby case. The second set simulates high-enthalpy steam flow conditions and is applied to the Brejoes case.

Table 1: Boundary conditions.

Parameter	**Whitby**	**Brejoes**
Rock heat conduction, κ_r [kJ/m/day/K]	165	150
Rock heat capacity, C_r [kJ/m^3/K]	2500	2200
Initial pressure, p_0 [bar]	500	100
Initial temperature, T_0 [K]	423.15	583.15
Injection rate, Q_{inj} [m^3/day]	1000	300
Injection temperature, T_{inj} [K]	303.15	308.15
Production bottom hole pressure, p_{prod} [bar]	475	100

Table 2: Reservoir and simulation parameters.

Parameter	**Whitby**	**Brejoes**
Matrix permeability, k_{mat} [mD]	1e−3	1e−2
Matrix porosity, ϕ_{mat} [-]	0.3	0.04
Fracture permeability, k_{frac} [mD]	8.3e7	7.5e6
Fracture porosity, ϕ_{frac} [-]	1	1
Length domain, L_x [m]	1050	700
Width domain, L_y [m]	1050	350
Simulation time, t [days]	10950	10950

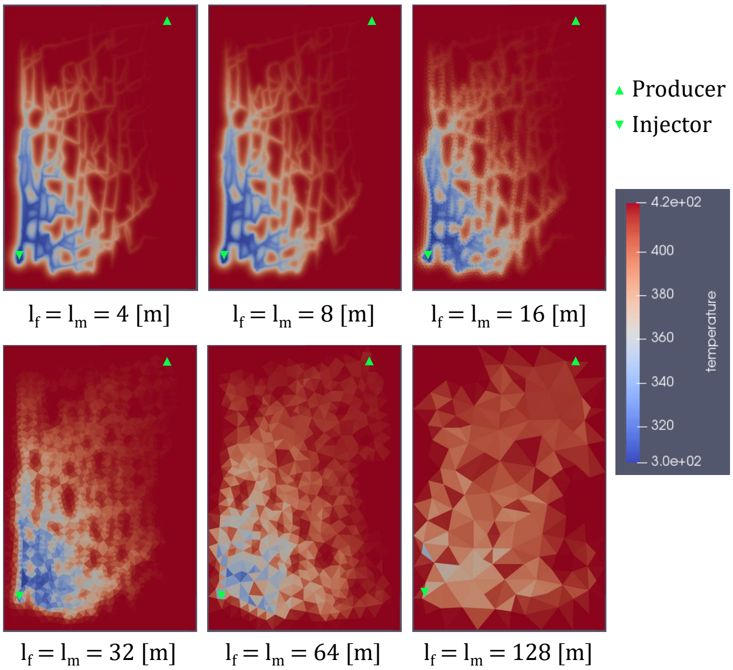


Figure 14. Temperature distribution as a function of preprocessing and meshing accuracy for the optimal strategy after 3150 [days] (Whitby network).

The temperature field after 3150 [days] of simulation for the Whitby network is presented in Figure 14. The temperature is reduced near the injection point from the initial 423.15 [K] to the injection temperature of 303.15 [K]. Fluid flow primarily happens through the fractures, hence the largest temperature variations occur closer to the fractures. This is more apparent in the finer models (i.e., smaller l_m). Larger diffusion of the temperature profile is observed for increasing l_m. The main fracture pattern becomes invisible at $l_m = 64$ [m]. The temperature distribution for the Brejoes network is shown in Figure 15. In terms of temperature distribution, a comparable trend was observed w.r.t. to the Whitby network. The water saturation field is shown in Figure 16 after 150 days of simulation. Accurate representation of the water saturation is more sensitive to the resolution than temperature.

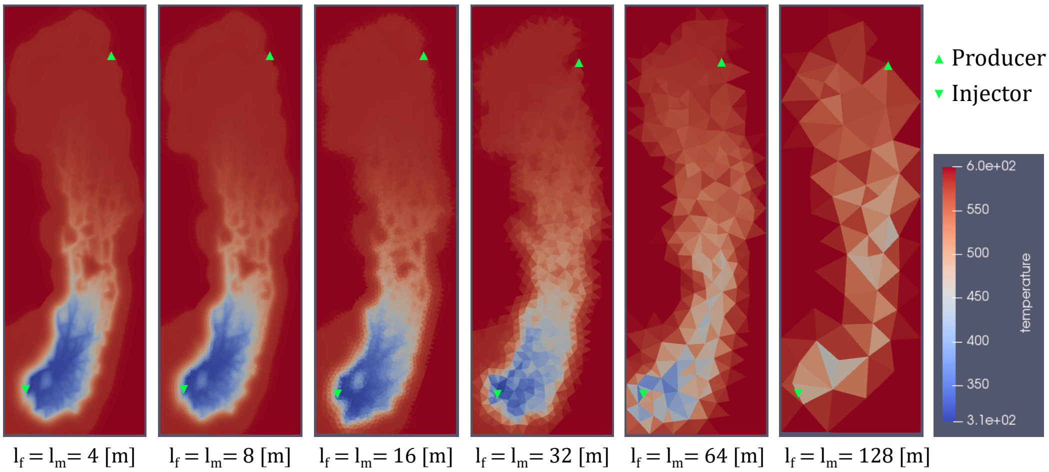


Figure 15. Temperature distribution as a function of preprocessing and meshing accuracy for the optimal strategy after 3150 [days] (Brejoes network).

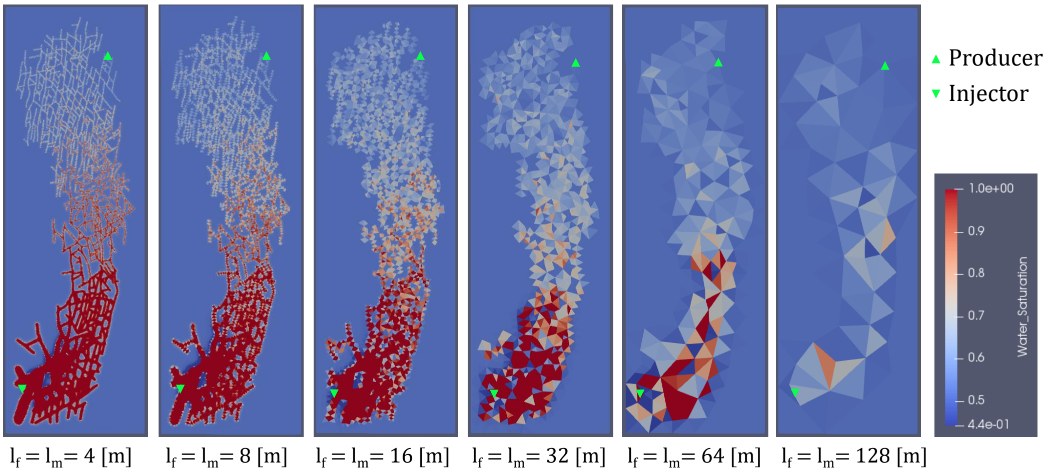


Figure 16. Water saturation distribution as a function of preprocessing and meshing accuracy for the optimal strategy after 150 [days] (Brejoes network).

The energy rate and temperature profile at the production well showed similar behavior; therefore, only the temperature profiles are shown in Figure 17. A commonly used metric to analyze the flow behavior of geothermal systems is the doublet lifetime. The lifetime is typically reached when the water temperature at the production well has decreased with 10-20% of the difference between initial and injection temperature. The optimal strategy (i.e., $l_f = l_m$) in the Whitby network starts deviating from the finer scales at $l_f = 32$ [m], particularly the lifetime is reduced by 670 [days]. From $l_f = 64$ [m] the deviation becomes more significant, notably a 2700 [days] difference in lifetime due to early breakthrough of the cold water. At $l_f = 128$ [m], the response does not resemble the finer scales specifically the lifetime is reduced to 500 [days] due to almost instant cold water breakthrough.

The clean strategy (i.e., $l_f = 1$ and $l_m = l_m$) shows an analogous result to the optimal strategy for the small l_m. For larger l_m the result of the clean strategy is significantly closer to the finer scales; particularly, there is no deviation in breakthrough times between the scales. This is expected since the fracture network is not changing (i.e., $l_f = 1$ for all simulations) with increasing l_m. Therefore, no changes in connectivity or the path from injector to producer occur, which is important in this tight fractured reservoir setting. Meshing artifacts in the clean strategy increase the number of control volumes for larger l_m, contributing to small changes across the scales (see Table 3). The difference between the clean and optimal strategy (i.e., $l_f = l_m$) for small l_m (≤ 32) in terms of flow-response is negligible; however, the performance of the optimal strategy is significantly better.

A more significant deviation in Brejoes temperature profile for the optimal case is observed. This is in line with the other observations. This pattern is observed in the angle distribution in the previous section (see Figure 8). Furthermore, Brejoes fracture density is larger (i.e., spacing between fractures is shorter), which leads to a more diffused and less complex temperature distribution. The large connectivity also means a shorter and highly conductive path from injector to producer, resulting in an early cold-water breakthrough.

3.2.2 High permeable matrix (low permeable fracture)

Fractures are often seen as high-permeable conduits, but there are cases where fractures or other discontinuities end up blocking fluid-flow (Gale et al., 2004). The test case presented here consists of a single phase-steady state simulation in a reservoir with a high permeable matrix and low permeable fractures. A small gap between the fractures exists in the middle of the domain $(x, y) = (500, 500)$. Since the gap exists below the meshing resolution accuracy, the preprocessing algorithm connects the fractures. In the case of no aperture correction, the flow can only go around the fractures. With aperture correction, the preprocessed model correctly represents a similar flow pattern as in the high fidelity model with streamlines passing through the middle of the domain.

3.2.3 Low connectivity (variable aperture)

The final test case consists of the variable aperture model applied to the Whitby fracture network (as seen in Figure 2). Since most of the high permeable fractures (N-S oriented) are connected through low permeable fracture (E-W oriented), this variable aperture model results in a low connectivity fracture model. A single-phase tracer simulation shows that the aperture correction is able to preserve the low connectivity of the network up to the numerical accuracy corresponding to the higher numerical dispersion in the coarser model. The early breakthrough of the tracer without aperture correction is obvious, according to Figure 2.

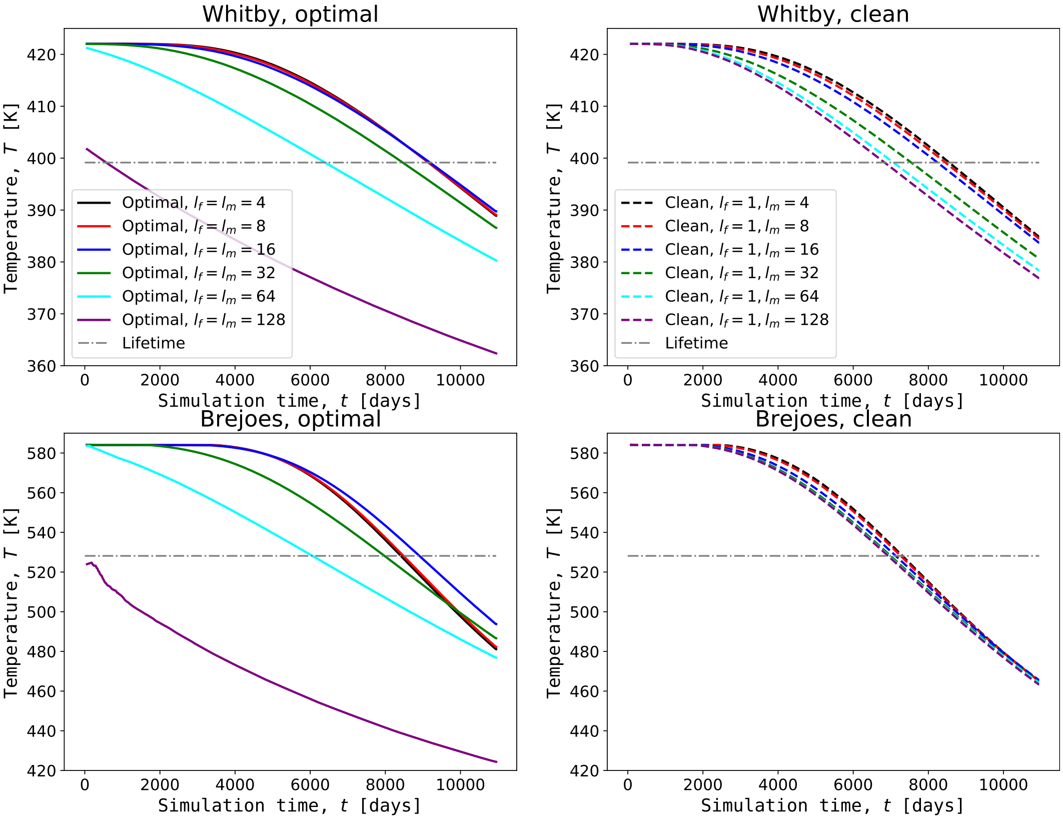
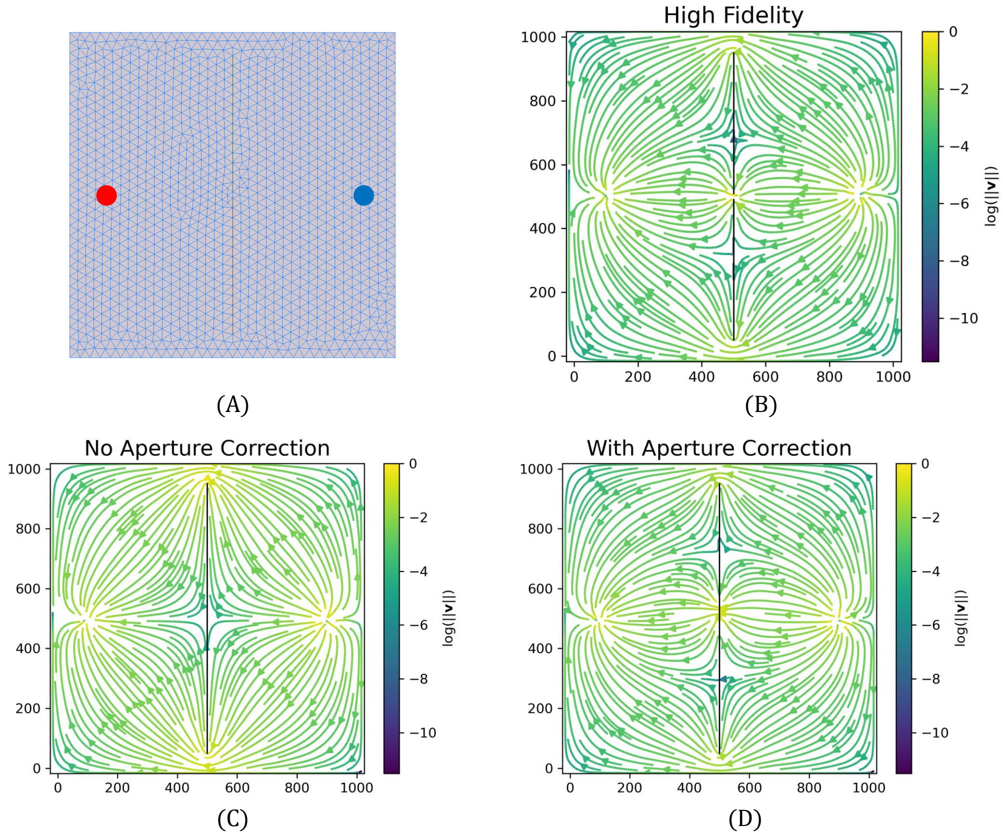


Figure 17. The temperature at the production well over time for optimal (left column) and clean (right column) preprocessing strategies for both the Whitby (top row) and Brejoes (bottom row) networks. Substantial deviation for large $l_f = l_m$ in the optimal strategy was observed. This does not happen in the clean strategy. This is because the fracture network is unchanged while the mesh is coarsened. This also causes the number of control volumes to remain considerable even for large l_m thereby reducing the numerical diffusion (see Table 3 and Table 4).



(A) (B)

(C) (D)

Figure 18. Single phase steady-state simulation of a high-permeable matrix and low permeable fractures. There is a small gap between the fractures in the high fidelity model allowing for fluid to pass. Not applying any fracture correction blocks the flow, while the aperture correction accurately depicts the high fidelity behavior.

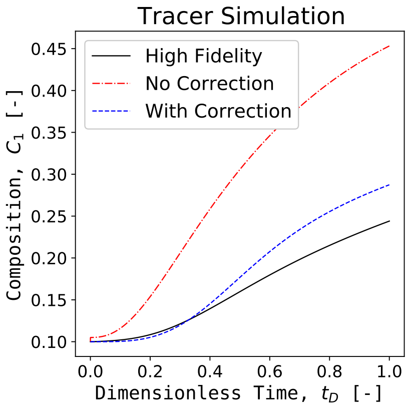


Figure 19. Single phase tracer simulation on the Whitby network with the variable aperture model depicted in Figure 2. A better match in breakthrough times is obtained after using the aperture correction.

Table 3. Numerical performance Whitby simulations. N_{blocks} corresponds to the total number of control volumes, N_{fracs} to the number of fracture control volumes, N_{newt} to the number of Newton-iterations, N_{lin} to the number of linear-iterations, and T_{CPU} to the total simulation time. l_f refers to the preprocessing accuracy, and l_m refers to the meshing accuracy.

	N_{blocks}	N_{fracs}	N_{newt}	N_{lin}	T_{CPU} [s]
Clean ($l_f = 1$, $l_m = 4$)	91,780	6,800	3,543	53,210	4,159
Clean ($l_f = 1$, $l_m = 8$)	41,119	4,311	3,277	46,830	1,290
Clean ($l_f = 1$, $l_m = 16$)	24,044	3,152	3,199	40,566	538
Clean ($l_f = 1$, $l_m = 32$)	22,879	2,841	3,112	39,667	364
Clean ($l_f = 1$, $l_m = 64$)	20,142	2,824	3,087	39,340	400
Clean ($l_f = 1$, $l_m = 128$)	20,222	2,824	3,085	38,903	422
Optimal ($l_f = l_m = 4$)	80,672	6,362	3,436	50,573	4,079
Optimal ($l_f = l_m = 8$)	26,553	3,363	2,890	37,988	813
Optimal ($l_f = l_m = 16$)	8,718	1,594	2,680	32,600	196
Optimal ($l_f = l_m = 32$)	2,417	563	2,533	27,434	53
Optimal ($l_f = l_m = 64$)	605	147	2,395	23,119	18
Optimal ($l_f = l_m = 128$)	166	32	2,403	17,147	6

3.3 Numerical performance high enthalpy

The numerical performance of the two strategies can be found in Table 3 and Table 4 for Whitby and Brejoes, respectively. No timestep cuts are observed in both strategies for the Whitby simulations. However, several timestep cuts were observed in both strategies for the Brejoes simulations. This is reflected in the larger amount of nonlinear and linear iterations. The convergence issues can be explained by the combination of complex two-phase physics (steam condensation) and DFM in the case of high-enthalpy two-phase flow. A more sophisticated nonlinear strategy can be utilized to limit the timestep cuts (Wang & Voskov, 2019), but the main goal of this study is to have a fair comparison between the two preprocessing strategies for the conventional nonlinear solver.

It is observed that the optimal strategy shows a better convergence in both networks. A reduction in nonlinear iterations of roughly 20% for the coarse models in the Whitby simulations is observed. In the Brejoes simulations, this reduction is almost 45%. The total CPU time for the optimal strategy in the Brejoes network increases slightly at the coarsest level due to a higher number of control volumes when the coarsest strategy is applied since the scale of the cleaning mainly constrains the meshing. For the optimal strategy at the coarsest scale, the simulation time is primarily dominated by the linearization step (i.e., construction of the operators for the OBL method) and therefore doesn't reduce below 32 seconds.

The number of control volumes N_{blocks} in the clean strategy does not drop below 48-50 thousand for Whitby and 22-26 thousand for Brejoes. This is because the fracture network, at the preprocessing accuracy of $l_f = 1$, is too complex for the meshing software at large l_m. The result is a substantial amount of elements with low mesh quality (see Figure 13) and no further reduction in N_{blocks} with increasing l_m. This significantly increases the computational time for the clean strategy when compared with the optimal strategy. For example, at $l_f = l_m = 32$ the optimal strategy only takes 14.6% of the clean strategy simulation time. However, this comes at the cost of a less accurate simulation response (see Figure 17).

Table 4. Numerical performance Brejoes simulations. N_{blocks} corresponds to the total number of control volumes, N_{fracs} to the number of fracture control volumes, N_{newt} to the number of Newton-iterations, N_{lin} to the number of linear-iterations, and T_{CPU} to the total simulation time~~, T_{ls} the total linear-solver time, and T_{lz} the total lineraziation time (constructing operators)~~. l_f refers to the preprocessing accuracy, and l_m refers to the meshing accuracy.

	N_{blocks}	N_{fracs}	N_{newton}	N_{linear}	T_{CPU} [s]
Clean ($l_f = 1$, $l_m = 4$)	157,105	8,079	6,970	163,388	6,803
Clean ($l_f = 1$, $l_m = 8$)	58,912	4,682	4,947	87,940	1,607
Clean ($l_f = 1$, $l_m = 16$)	30,739	3,035	5,129	80,568	856
Clean ($l_f = 1$, $l_m = 32$)	22,918	2,402	4,784	77,690	766
Clean ($l_f = 1$, $l_m = 64$)	24,955	2,233	5,038	78,795	618
Clean ($l_f = 1$, $l_m = 128$)	26,127	2,211	4,851	75,687	551
Optimal ($l_f = l_m = 4$)	150,566	7,852	4,354	108,073	3,909
Optimal ($l_f = l_m = 8$)	46,811	4,115	3,308	52,374	564
Optimal ($l_f = l_m = 16$)	15,139	2,093	2,979	38,458	167
Optimal ($l_f = l_m = 32$)	4,899	967	2,747	27,698	50
Optimal ($l_f = l_m = 64$)	1,471	371	2,632	20,254	32
Optimal ($l_f = l_m = 128$)	400	122	2,562	14,203	34

4 Discussion

The existing preprocessing strategies described in the literature only implicitly resolve the fracture segments that intersect at a small angle via node merging. We augment this with an extra step where all the low-angle intersections are explicitly resolved and improve the volume distribution, mesh quality, and the convergence of subsequent numerical simulation. Furthermore, we presented an aperture correction technique that allows handling of realistic aperture distributions and low connected fracture networks. We also contribute a comprehensive investigation of the geometry and topology changes as a function of discretization accuracy and its effect on the dynamic reservoir behavior. Next, we discuss static and dynamic results of our study and give our recommendations.

4.1 Topology

The inherent bias of artificial connectivity in the coarser models is evident in the static analysis. Especially the topology is sensitive to subtle changes in the fracture network. The preprocessing method does seem to converge given that the distance in the ternary topology diagram appears to decrease with decreasing l_f (except for two jumps in the Brejoes topology data for $l_f = 1$ and $l_f = 128$ [m]).

The large deviation from the raw topology can be explained through several points. Manual interpretation is usually made in some software (e.g., QGIS) or on the image directly. Every fracture is interpreted as a line, and two points are connected, particularly the beginning- and end-point of the fracture. Even if the interpreter meant for the two fractures to abut against each other, beginning- or end-points are rarely placed exactly on top of the existing line. The computer processing interprets the point as I- or X-node, while the interpreter meant the node to be a Y-node. This can be omitted if some snipping tool during the interpretation is used or a semi-automated (Vasuki et al., 2014) or fully automated (Prabhakaran et al., 2019) interpretation method. However, this is not always the case as shown for two networks chosen in this study.

The other problem is the scale of the image. The Brejoes data set has a huge resolution (20 mm/pixel) (Prabhakaran et al., 2019). It can be argued that you would roughly need 15-25 pixels to be sure about the interaction of two or more fractures due to shading, contrast, and other optical effects in the image. Considering this, it would mean that intersection and abutment relationships cannot be interpreted at a scale smaller than 300-500 [mm] (for this particular image).

Furthermore, the image shows a 2D representation of the fracture network. In 3D, fractures are represented by planes. Any deviation from perfectly vertical planes would increase the chance of nodes classified as I-nodes turning into Y-nodes. All of this leads to the argument that the raw network data should not be used in the topological assessment of fracture networks. However, a small cleaning should be applied for the analysis to provide meaningful results. ~~This observation is similar to ? (?) (refer to Auke's paper here with the image segmentation of grains).~~

4.2 Fluid flow

As shown in Figure 14 and 15, the predictions on flow response do not seem to be affected by small details in the fracture network. However, they are substantially different after successive coarsening (i.e., increasing l_c). The main reason for the earlier water breakthrough observed in Figure 17 can be attributed to an increase in connectivity of the fracture network (see also Figure 9). Furthermore, the shortest flow path through the fracture network from the injector to the producer is significantly reduced in the coarser models; hence the cold water arrives earlier. Finally, since the volume of the fractures is unchanged, even if two fracture segments are merged, the fluid velocity through a merged fracture is higher for the same injection rate. All of these things affect the time the water has to heat up (i.e., recharge) and reduce the breakthrough time of the cold water in the coarser models.

Even without using flow-based upscaling when coarsening the mesh (i.e., increasing l_m), the flow-response for the coarser models remains accurate (up to $l_m = 32$ for the Whitby simulation and up to $l_m = 16$ for Brejoes). This implies that for a fraction of the computational time of the high-fidelity model (i.e., 1.3% for Whitby and 4.3% for Brejoes), we are still able to obtain a representative flow and heat transfer for this complex physical process. This opens up avenues for replacing effective media models in common optimization and uncertainty quantification practices, such as Arnold et al. (2016) and Spooner et al. (2019), with more accurate DFM models (de Hoop & Voskov, 2021).

The main idea is that adding a fracture to an already connected network is not necessarily a problem. Connecting whole clusters that were not previously connected can pose a significant issue and significantly affects the flow and heat transfer in the reservoir. To remedy this, we added a novel aperture correction that penalizes connecting fracture segments that do not already share a connection. The tracer simulation applied to the Whitby fracture network with variable aperture distribution significantly improves the match between the coarser representation and the high-fidelity model in terms of breakthrough time (see Figure 19). Furthermore, the aperture correction allows us to deal with coarsening of sealing fractures that potentially block fluid-flow, as observed in the synthetic test case in Figure 18.

~~A possible addition to the preprocessing method is keeping track of the volume changes of the fractures. When two fracture segments are merged, the cumulative volume could be recorded. The main benefit is that the fracture volume is preserved. The downside is that the hydraulic conductivity (permeability) of the fracture is dependent on the square of the aperture. This means that a cumulative volume (i.e., adding the aperture of the two segments that are merged) results in a doubled hydraulic conductivity.~~

~~The networks differ mainly in terms of angles and C_L. However, it is important to touch on the similarity between the two fracture networks used in this study: both are quite well connected. This, in combination with the observations on the earlier water breakthrough for coarser models, begs the question: how does the preprocessing algorithm work on networks that are poorly connected? This question is part of our future work. The ultimate goal is to use the presented approach for an efficient and robust uncertainty quantification procedure in fractured reservoirs of any connectivity.~~

~~When you have a large number of disconnected clusters, as long as the l_f is below the smallest distance between those clusters and below the average spacing of the fractures that predominantly affect the fluid flow, the preprocessing should be accurate even for very coarse models. Whenever disconnected clusters are within a small distance of each other, this becomes more difficult. A smaller l_f should be used since clusters will become connected while, in reality, they are not. Another remedy for this problem would be disconnecting clusters that become connected after preprocessing. This can be achieved by accurate bookkeeping (i.e., recording before preprocessing to which cluster a certain node belongs and observing how this evolves when running the algorithm). The issue of not preserving average spacing for very coarse models and thereby greatly altering the fractures' orientation is illustrated in Figure 7 and 8.~~

~~In our future work, we have generated a large ensemble of varying connectivity for future work and observe how the flow-response accuracy w.r.t. fine-scale changes with increasing l_f to shed light on this issue.~~

4.3 Application and recommendations

It seems from the study presented in this paper that the flow-response is less sensitive to changes in the fracture network than initially thought. The orientation of the fractures (i.e., angle distribution) is also less sensitive than the topology. This could serve as a recommendation to geologists and modelers that the scale and complexity at which the data is collected and the models are constructed is unnecessarily refined. It would save time and improve the ambiguity of our models to set a certain interpretation scale at which you can be certain of the intersection and abutment relationships before making the interpretation.

The preprocessing method effectively extracts the backbone of a complex fracture network. Therefore, it can be used to extract the main pattern of the network and might be useful when generating training images for algorithms such as Bruna et al. (2019).

~~The computational time of the preprocessing is insignificant to the simulation time of fine-scale (especially after parallelization, which is already implemented and will be described in our future work). Even more so if new functionality is utilized, such as the Numba Python package where functions are translated into machine code before executing the script to speed up computations. The significant speed-up in computational time for the coarser optimal preprocessing strategy allows for the utilization in uncertainty quantification of fractured reservoirs. Since the main reason in uncertainty quantification is understanding the stochastic response instead of having a single very accurate simulation response.~~

5 Conclusion

This study demonstrates a strategy to simplify complex fracture networks in terms of flow response based on a robust preprocessing approach using graph theory. We show that using raw fracture data for topological analysis and dynamic modeling is unwise and that some preprocessing should be applied to investigate the patterns that exist in the studied network. Our method simplifies the topology of the fracture network by merging fracture nodes (i.e., vertices) within a certain radius. Consequently, this amounts to taking the union of the incidence matrix's rows of each vertex, thereby preserving all the connectivity within the fracture network. Furthermore, it explicitly removes problematic fracture intersections that occur at an angle below a certain threshold. Finally, our frameworks extends the current preprocessing methods, such as Mustapha and Mustapha (2007) and Karimi-Fard and Durlofsky (2016), by taking into consideration an aperture correction when vertices are merged to better preserve the original connectivity and handle heterogeneous aperture distributions.

Our preprocessing framework can create a fully conformal uniformly distributed grid for a given realistic fracture network with variable aperture at the required level of accuracy. The changes introduced by the method are analyzed in terms of geometry (i.e., angle distribution of the fracture network), meshing results (i.e., volume and quality of the elements), and dynamic response of the reservoir when subjected to geothermal high-enthalpy production conditions. Results are analysed for two realistic fracture networks based on outcrop studies, a synthetic case with sealing fractures, and a variable aperture model. Topology is more affected by the preprocessing than the geometry and flow response in studied networks. ~~The performance of our method in low connectivity networks will be a part of future research.~~

Uncertainty quantification relies on a large number of numerical simulations. The presented method decreases the computational complexity of DFM models. Therefore, our approach opens up avenues for using efficient DFM models with similar computational complexity as embedded-DFM (EDFM) and even Dual-Porosity models while accurately capturing the discrete nature of fracture networks for uncertainty quantification and history matching purposes. This is especially true for the optimal preprocessing strategy where cleaning and optimizing the fracture network, including treatment of intersections, node merging, and straightening, are combined.

The open-source computational framework performing all the preprocessing stages can be found at `https://github.com/MakeLikePaperrr/Fracture-Preprocessing-Code`.

6 Appendix: various algorithms for DFN preprocessing

Algorithm 1 Construct graph

```
$V = \{\}$
$n = 0$
for $(x_i, y_i, x_j, y_j) \in \mathcal{F}$ do
    if $(x_i, y_i) \notin V$ then
        $V = V \cup (x_i, y_i)$
        $n \mathrel{+}= 1$

    if $(x_j, y_j) \notin V$ then
        $V = V \cup (x_j, y_j)$
        $n \mathrel{+}= 1$

$B = \text{zeros}(n, m)$
for $(x_i, y_i) \in V$ do
    ids = $\text{find}(\forall(x_i, y_i) \in \mathcal{F}(\cdot, [1, 2]) \wedge \forall(x_i, y_i) \in \mathcal{F}(\cdot, [3, 4]))$
    $B(i, \text{ids}) = 1$

$D = \text{diag}(B\mathbf{1}_{m\times 1})$
$A = BB^T - D$
$L = D - A$
```

Algorithm 2 Partition segments

```
$m_{\text{new}} = \sum_i^m \max(1, \text{round}(l_i/l_f))$
$\mathcal{F}_{\text{new}} = \text{zeros}(m_{\text{new}}, 4)$
count = 1
for $k \in O_{\text{segm}}$ do
    $m_k = \max(1, \text{round}(l_k/l_f))$
    ids $= [1, \ldots, m_k]$
    $\mathcal{F}_{\text{new}}(\text{count} : (\text{count} + m_k), 1) = \mathcal{F}(k, 1) + (\text{ids} - 1)/m_k(\mathcal{F}(k, 3) - \mathcal{F}(k, 1))$
    $\mathcal{F}_{\text{new}}(\text{count} : (\text{count} + m_k), 2) = \mathcal{F}(k, 2) + (\text{ids} - 1)/m_k(\mathcal{F}(k, 4) - \mathcal{F}(k, 2))$
    $\mathcal{F}_{\text{new}}(\text{count} : (\text{count} + m_k), 3) = \mathcal{F}(k, 1) + \text{ids}/m_k(\mathcal{F}(k, 3) - \mathcal{F}(k, 1))$
    $\mathcal{F}_{\text{new}}(\text{count} : (\text{count} + m_k), 4) = \mathcal{F}(k, 2) + \text{ids}/m_k(\mathcal{F}(k, 4) - \mathcal{F}(k, 2))$
    count $\mathrel{+}= m_k$

$O_{\text{segm, new}} = [1, \ldots, m_{\text{new}}]$ //since $\mathcal{F}_{\text{new}}$ is already ordered now!
```

Algorithm 3 Determine order vertices

```
$B = B(\cdot, O_{\text{segm}})$ //order the columns of $B$
$O_{\text{vertices}}$ = zeros(n, 1)
count $= 0$
for $k = 1$ to $m$ do
    $(i, j) = \text{find}(B(\cdot, k) == 1)$
    if $i \notin O_{\text{vertices}}$ then
        count $+= 1$
        $O_{\text{vertices}}(\text{count}) = i$

    if $j \notin O_{\text{vertices}}$ then
        count $+= 1$
        $O_{\text{vertices}}(\text{count}) = j$

$\mathcal{X} = \mathcal{X}(O_{\text{vertices}}, \cdot)$ //sort vertices
$B = B(O_{\text{vertices}}, \cdot)$ //sort rows of incidence matrix accordingly
```

Algorithm 4 Node merging

```
$D_X = \text{pdist}(\mathcal{X})$ //pairwise symmetric $n \times n$ distance matrix for each vertex in $\mathcal{X}$
mergelist = zeros(n, 1)
for $k = 2$ to $n$ do
    $\text{id}_{\text{min}} = \min(\{d_{k,i} \in D_X \quad | \quad \forall i \in \mathbb{N}, \quad i < k\})$ //closest vertex already in domain
    if $D_X(k, \text{id}_{\text{min}}) < l_f/2$ then
        $\text{mergelist}(k) = \text{id}_{\text{min}}$
        $B(\text{id}_{\text{min}}, \cdot) = B(\text{id}_{\text{min}}, \cdot) \cup B(k, \cdot)$ //record new connections from node merging
        $B(k, \cdot) = 0$ //remove merged node from graph
        $D_X(k, \cdot) = \infty$ //reset distance from removed node
        $D_X(\cdot, k) = \infty$ //reset distance from removed node

$\text{mask} = \{i \in \mathbb{N} \quad | \quad \forall i \notin \text{mergelist}, \quad i \leq n\}$
$\mathcal{X} = \mathcal{X}(\text{mask})$
$n = \text{card}(\mathcal{X})$
$B = B(\text{mask}, \cdot)$
$B = B(\cdot, \mathbf{1}_{1\times n} B > 1)$ //remove “collapsed” edges
$B = \text{unique}(B, \text{'cols'})$ //remove overlapping edges
```

Algorithm 5 Straighten fractures

```
nodelist $= \{d_i \in D(G) \quad | \quad d_i == 2\}$
mergelistnodes $=$ zeros(n, 1)
mergelistsegms $=$ zeros(m, 1)
for $k \in$ nodelist do
    idsegms $= \text{nonzero}(B(k, \cdot))$
    $\mathbf{v}_1 = \mathcal{F}(\text{idsegms}(1), [1, 2]) - \mathcal{F}(\text{idsegms}(1), [3, 4])$
    $\mathbf{v}_2 = \mathcal{F}(\text{idsegms}(2), [1, 2]) - \mathcal{F}(\text{idsegms}(2), [3, 4])$
    dotproduct $= \min(1, \max(-1, \frac{\mathbf{v}_1^T \mathbf{v}_2}{||\mathbf{v}_1||||\mathbf{v}_2||}))$
    $\theta = \arccos(\text{dotproduct})180/\pi$
    if $\theta < \theta_{\text{tol}}$ then
        mergelistnodes $= k$
        $B(k, \cdot) = 0$
        mergelistsegms $=$ idsegms(2)
        idnodes $= \text{nonzero}(B(\cdot, \text{idsegms}(1))) \cup \text{nonzero}(B(\cdot, \text{idsegms}(2)))$
        $B(\cdot, \text{idsegms}(2)) = 0$
        $B(\text{idnodes} \neq k, \text{idsegms}(1)) = 1$

//$B$ and $\mathcal{X}$ are updated similarly to Algorithm 4 using "mergelistnodes" and
"mergelistsegms" for the removed vertices and edges respectively
```

Acknowledgments

The project was performed with a subsidy (reference TKI2017-07-UG) from the Ministry of Economic Affairs, National schemes EZ subsidies, Top sector Energy, carried out by the Netherlands Enterprise Agency. We also want to thank Mohammed Karimi-Fard for his insightful comments and suggestions regarding the preprocessing strategy. Finally, we want to thank Alexandros Daniilidis for his valuable discussion regarding the aperture correction procedure. The data and source code is available at `https://github.com/MakeLikePaperrr/Fracture-Preprocessing-Code`.

References

Acuna, J. A., & Yortsos, Y. C. (1995). Application of fractal geometry to the study of networks of fractures and their pressure transient. *Water Resources Research*, *31*(3), 527–540.

Arnold, D., Demyanov, V., Christie, M., Bakay, A., & Gopa, K. (2016). Optimisation of decision making under uncertainty throughout field lifetime: A fractured reservoir example. *Computers & Geosciences*, *95*, 123–139.

Baghbanan, A., & Jing, L. (2008). Stress effects on permeability in a fractured rock mass with correlated fracture length and aperture. *International journal of rock mechanics and mining sciences*, *45*(8), 1320–1334.

Balberg, I., & Binenbaum, N. (1983). Computer study of the percolation threshold in a two-dimensional anisotropic system of conducting sticks. *Physical Review B*, *28*(7), 3799.

Barenblatt, G. (1960). Basic concepts in the theory of seepage of homogeneous liquids in fissured rocks. *Prikl. Mat. Mekh.*, *24*(5), 852–864.

Berre, I., Doster, F., & Keilegavlen, E. (2019). Flow in fractured porous media: a review of conceptual models and discretization approaches. *Transport in Porous Media*, *130*(1), 215–236.

Bisdom, K., Bertotti, G., & Nick, H. M. (2016). A geometrically based method for predicting stress-induced fracture aperture and flow in discrete fracture networks. *AAPG Bulletin*, *100*(7), 1075–1097.

Bisdom, K., Nick, H., & Bertotti, G. (2017). An integrated workflow for stress and flow modelling using outcrop-derived discrete fracture networks. *Computers & Geosciences*, *103*, 21–35.

Boersma, Q., Athmer, W., Haege, M., Etchebes, M., Haukås, J., & Bertotti, G. (2020). Natural fault and fracture network characterization for the southern ekofisk field: A case study integrating seismic attribute analysis with image log interpretation. *Journal of Structural Geology*, *141*, 104197.

Boersma, Q., Prabhakaran, R., Bezerra, F. H., & Bertotti, G. (2019). Linking natural fractures to karst cave development: a case study combining drone imagery, a natural cave network and numerical modelling. *Petroleum Geoscience*, *25*(4), 454–469.

Boersma, Q. D., Bruna, P. O., De Hoop, S., Vinci, F., Tehrani, A. M., & Bertotti, G. (2021). The impact of natural fractures on heat extraction from tight triassic sandstones in the west netherlands basin: a case study combining well, seismic and numerical data. *Netherlands Journal of Geosciences*, *100*.

Bollobás, B. (2013). *Modern graph theory* (Vol. 184). Springer Science & Business Media.

Bruna, P.-O., Straubhaar, J., Prabhakaran, R., Bertotti, G., Bisdom, K., Mariethoz, G., & Meda, M. (2019). A new methodology to train fracture network simulation using multiple-point statistics. *Solid Earth*, *10*(2), 537–559.

Csardi, G., Nepusz, T., et al. (2006). The igraph software package for complex network research. *InterJournal, complex systems*, *1695*(5), 1–9.

de Hoop, S., & Voskov, D. (2021). Fast and robust scheme for uncertainty quantification in naturally fractured reservoirs. In *Spe reservoir simulation conference.*

Flemisch, B., Berre, I., Boon, W., Fumagalli, A., Schwenck, N., Scotti, A., ... Tatomir, A. (2018). Benchmarks for single-phase flow in fractured porous media. *Advances in Water Resources*, *111*, 239–258.

Gale, J. F., Laubach, S. E., Marrett, R. A., Olson, J. E., Holder, J., & Reed, R. M. (2004). Predicting and characterizing fractures in dolostone reservoirs: Using the link between diagenesis and fracturing. *Geological Society, London, Special Publications*, *235*(1), 177–192.

Gallyamov, E., Garipov, T., Voskov, D., & Van den Hoek, P. (2018). Discrete fracture model for simulating waterflooding processes under fracturing conditions. *International Journal for Numerical and Analytical Methods in Geomechanics*, *42*(13), 1445–1470.

Garipov, T., Karimi-Fard, M., & Tchelepi, H. (2016). Discrete fracture model for coupled flow and geomechanics. *Computational Geosciences*, *20*(1), 149–160.

Geiger, S., & Matthäi, S. (2014). What can we learn from high-resolution numerical simulations of single-and multi-phase fluid flow in fractured outcrop analogues? *Geological Society, London, Special Publications*, *374*(1), 125–144.

Geuzaine, C., & Remacle, J.-F. (2009). Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, *79*(11), 1309–1331.

Gureghian, A. B. (1975). A study by the finite-element method of the influence of fractures in confined aquifers. *Society of Petroleum Engineers Journal*, *15*(02), 181–191.

Hajibeygi, H., Karvounis, D., & Jenny, P. (2011). A hierarchical fracture model for the iterative multiscale finite volume method. *Journal of Computational Physics*, *230*(24), 8729–8743.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... Oliphant, T. E. (2020, September). Array programming with NumPy. *Nature*, *585*(7825), 357–362. Retrieved from `https://doi.org/10.1038/s41586-020-2649-2` doi: 10.1038/s41586-020-2649-2

Helmig, R., et al. (1997). *Multiphase flow and transport processes in the subsurface:*

a contribution to the modeling of hydrosystems. Springer-Verlag.

HosseiniMehr, M., Piguave Tomala, J., Vuik, C., & Hajibeygi, H. (2020). Projection-based embedded discrete fracture model (pedfm) on corner-point grid geometry for subsurface flow and geothermal modeling.. doi: 10.3997/2214-4609.202035245

Houben, M., Hardebol, N., Barnhoorn, A., Boersma, Q., Carone, A., Liu, Y., … Drury, M. (2017). Fluid flow from matrix to fractures in early jurassic shales. *International Journal of Coal Geology*, *175*, 26–39.

Jung, A., Fenwick, D. H., & Caers, J. (2013). Training image-based scenario modeling of fractured reservoirs for flow uncertainty quantification. *Computational Geosciences*, *17*(6), 1015–1031.

Kala, K., & Voskov, D. (2020). Element balance formulation in reactive compositional flow and transport with parameterization technique. *Computational Geosciences*, *24*(2), 609-624. doi: 10.1007/s10596-019-9828-y

Karimi-Fard, M., & Durlofsky, L. J. (2016). A general gridding, discretization, and coarsening methodology for modeling flow in porous formations with discrete geological features. *Advances in water resources*, *96*, 354–372.

Karimi-Fard, M., Durlofsky, L. J., & Aziz, K. (2004). An efficient discrete-fracture model applicable for general-purpose reservoir simulators. *SPE journal*, *9*(02), 227–236.

Karimi-Fard, M., Gong, B., & Durlofsky, L. J. (2006). Generation of coarse-scale continuum flow models from detailed fracture characterizations. *Water resources research*, *42*(10).

Khait, M., & Voskov, D. (2018a). Adaptive parameterization for solving of thermal/compositional nonlinear flow and transport with buoyancy. *SPE Journal*, *23*, 522-534. doi: 10.2118/182685-PA

Khait, M., & Voskov, D. (2018b). Adaptive parameterization for solving of thermal/compositional nonlinear flow and transport with buoyancy. *SPE Journal*, *23*(2), 522-534. doi: 10.2118/182685-pa

Khait, M., & Voskov, D. (2018c). Operator-based linearization for efficient modeling of geothermal processes. *Geothermics*, *74*, 7-18. doi: 10.1016/j.geothermics.2018.01.012

Khait, M., & Voskov, D. V. (2017). Operator-based linearization for general purpose reservoir simulation. *Journal of Petroleum Science and Engineering*, *157*, 990–998.

Koohbor, B., Fahs, M., Hoteit, H., Doummar, J., Younes, A., & Belfort, B. (2020). An advanced discrete fracture model for variably saturated flow in fractured porous media. *Advances in Water Resources*, *140*, 103602.

Koudina, N., Garcia, R. G., Thovert, J.-F., & Adler, P. (1998). Permeability of three-dimensional fracture networks. *Physical Review E*, *57*(4), 4466.

Li, L., & Lee, S. H. (2008). Efficient field-scale simulation of black oil in a naturally fractured reservoir through discrete fracture networks and homogenized media. *SPE Reservoir evaluation & engineering*, *11*(04), 750–758.

Li, L., & Voskov, D. (2021). A novel hybrid model for multiphase flow in complex multi-scale fractured systems. *Journal of Petroleum Science and Engineering*, *203*, 108657.

Li, X., & Li, D. (2019). A numerical procedure for unsaturated seepage analysis in rock mass containing fracture networks and drainage holes. *Journal of Hydrology*, *574*, 23–34.

Lyu, X., Khait, M., & Voskov, D. (2021). Operator-based linearization approach for modelling of multiphase flow with buoyancy and capillarity. *SPE Journal*. doi: 10.2118/205378-PA

Lyu, X., Voskov, D., & Rossen, W. R. (2021). Numerical investigations of foam-assisted co_2 storage in saline aquifers. *International journal of greenhouse gas control*.

Mallison, B. T., Hui, M.-H., & Narr, W. (2010). Practical gridding algorithms for discrete fracture modeling workflows. In *Ecmor xii-12th european conference on the mathematics of oil recovery* (pp. cp–163).

Manzocchi, T. (2002). The connectivity of two-dimensional networks of spatially correlated fractures. *Water Resources Research*, *38*(9), 1–1.

Maryška, J., Severỳn, O., & Vohralík, M. (2005). Numerical simulation of fracture flow with a mixed-hybrid fem stochastic discrete fracture network model. *Computational Geosciences*, *8*(3), 217–234.

Moeck, I. S. (2014). Catalog of geothermal play types based on geologic controls. *Renewable and Sustainable Energy Reviews*, *37*, 867–882.

Moinfar, A., Narr, W., Hui, M.-H., Mallison, B. T., & Lee, S. H. (2011). Comparison of discrete-fracture and dual-permeability models for multiphase flow in naturally fractured reservoirs. In *Spe reservoir simulation symposium.*

Mustapha, H., & Dimitrakopoulos, R. (2011). Discretizing two-dimensional complex fractured fields for incompressible two-phase flow. *International Journal for Numerical Methods in Fluids*, *65*(7), 764–780.

Mustapha, H., & Mustapha, K. (2007). A new approach to simulating flow in discrete fracture networks with an optimized mesh. *SIAM Journal on Scientific Computing*, *29*(4), 1439–1459.

Nejadi, S., Trivedi, J. J., & Leung, J. (2017). History matching and uncertainty quantification of discrete fracture network models in fractured reservoirs. *Journal of Petroleum Science and Engineering*, *152*, 21–32.

Novikov, A., Voskov, D., Khait, M., Hajibeygi, H., & Jansen, J. (2021). A collocated finite volume scheme for high-performance simulation of induced seismicity in geo-energy applications.. doi: 10.2118/203903-MS

Prabhakaran, R., Bruna, P.-O., Bertotti, G., & Smeulders, D. (2019). An automated fracture trace detection technique using the complex shearlet transform. *Solid Earth*, *10*(6), 2137–2166.

Pruess, K., & Narasimhan, T. (1982). On fluid reserves and the production of superheated steam from fractured, vapor-dominated geothermal reservoirs. *Journal of Geophysical Research: Solid Earth*, *87*(B11), 9329–9339.

Reichenberger, V., Jakobs, H., Bastian, P., & Helmig, R. (2006). A mixed-dimensional finite volume method for two-phase flow in fractured porous media. *Advances in water resources*, *29*(7), 1020–1036.

Sanderson, D. J., & Nixon, C. W. (2015). The use of topology in fracture network characterization. *Journal of Structural Geology*, *72*, 55–66.

Sanderson, D. J., Peacock, D. C., Nixon, C. W., & Rotevatn, A. (2019). Graph theory and the analysis of fracture networks. *Journal of Structural Geology*, *125*, 155–165.

Spielman, D. A. (2010). Algorithms, graph theory, and linear equations in laplacian matrices. In *Proceedings of the international congress of mathematicians 2010 (icm 2010) (in 4 volumes) vol. i: Plenary lectures and ceremonies vols. ii–iv: Invited lectures* (pp. 2698–2722).

Spooner, V., Geiger, S., & Arnold, D. (2019). Ranking fractured reservoir models using flow diagnostics. In *Spe reservoir simulation conference.*

Spooner, V., Geiger, S., & Arnold, D. (2021). Dual-porosity flow diagnostics for spontaneous imbibition in naturally fractured reservoirs. *Water Resources Research*, e2020WR027775.

Ţene, M., Bosma, S. B., Al Kobaisi, M. S., & Hajibeygi, H. (2017). Projection-based embedded discrete fracture model (pedfm). *Advances in Water Resources*, *105*, 205–216.

Vasuki, Y., Holden, E.-J., Kovesi, P., & Micklethwaite, S. (2014). Semi-automatic mapping of geological structures using uav-based photogrammetric data: An image analysis approach. *Computers & Geosciences*, *69*, 22–32.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., … SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. doi: 10.1038/s41592-019-0686-2

Voskov, D. V. (2017). Operator-based linearization approach for modeling of multiphase multi-component flow in porous media. *Journal of Computational Physics*, *337*, 275–288.

Wagner, W., & Kretzschmar, H.-J. (2008). *International steam tables: Properties of water and steam based on the industrial formulation iapws-if97.* Springer-Verlag Berlin Heidelberg. Retrieved from `http://dx.doi.org/10.1007/978-3-540-74234-0` doi: 10.1007/978-3-540-74234-0

Wang, Y., de Hoop, S., Voskov, D., Bruhn, D., & Bertotti, G. (2021). Modeling of multiphase mass and heat transfer in fractured high-enthalpy geothermal systems with advanced discrete fracture methodology. *Advances in Water Resources*, 103985.

Wang, Y., & Voskov, D. (2019). High-enthalpy geothermal simulation with continuous localization in physics.. doi: https://pangea.stanford.edu/ERE/pdf/IGAstandard/SGW/2019/Wang4.pdf

Wang, Y., Voskov, D., Khait, M., & Bruhn, D. (2020a). An efficient numerical simulator for geothermal simulation: A benchmark study. *Applied Energy*, *264*, 114693.

Wang, Y., Voskov, D., Khait, M., & Bruhn, D. (2020b). An efficient numerical simulator for geothermal simulation: A benchmark study. *Applied Energy*, *264*. doi: 10.1016/j.apenergy.2020.114693

Warren, J., & Root, P. J. (1963). The behavior of naturally fractured reservoirs. *Society of Petroleum Engineers Journal*, *3*(03), 245–255.

Wellman, T. P., Shapiro, A. M., & Hill, M. C. (2009). Effects of simplifying fracture network representation on inert chemical migration in fracture-controlled aquifers. *Water Resources Research*, *45*(1).

West, D. B., et al. (2001). *Introduction to graph theory* (Vol. 2). Prentice hall Upper Saddle River.

Willems, C., & Nick, H. (2019). Towards optimisation of geothermal heat recovery: An example from the west netherlands basin. *Applied energy*, *247*, 582–593.

Wong, D. L. Y., Doster, F., Geiger, S., Francot, E., & Gouth, F. (2020). Fluid flow characterization framework for naturally fractured reservoirs using small-scale fully explicit models. *Transport in Porous Media*, *134*(2), 399–434.