Effective permeability in fractured reservoirs: Discrete fracture network simulations and percolation-based effective-medium approximation

by

Tolulope Agbaje

B.Tech., Federal University of Technology Akure, 2019

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Geology
College of Arts and Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2023

Approved by:

Major Professor
Behzad Ghanbarian

Copyright

Abstract

Fractured reservoirs are complex and multi-scale systems composed of matrix and fractures. Accordingly, modeling flow in such formations has been a great challenge. In this study, we investigated the effect of matrix and fracture network characteristics on the effective permeability (k_{eff}) in matrix-fracture systems of different sizes (L = 22.5, 30, 50, and 70 m). We generated fracture networks, embedded within a matrix of permeability, k_m = 10^{-18} m^2, using the discrete fracture network approach. Fracture length followed a truncated power-law distribution with exponent α = 1.5, 2.0, and 2.5, minimum fracture length l_{min} = 0.02 m, and maximum fracture length l_{max} = 20 m. The logarithmic ratio of fracture permeability (k_f) to matrix permeability (k_m) was set equal to 2, 4, and 6. We numerically simulated fluid flow to determine the k_{eff} for 36 sets of simulations (3 α × 3 log(k_f/k_m) × 4 L) over a wide range of fracture density ($0 \leq \rho \leq 1$). Numerical results showed that the impact of α and L on the k_{eff} became more significant as $\log(k_f/k_m)$ increased. Percolation-based effective-medium approximation (P-EMA) was fit to the simulated k_{eff}-ρ data, with an average R^2 = 0.99, and its parameters, ρ_c (critical fracture density) and t (scaling exponent), were optimized. We found ρ_c positively correlated to α and L, while t negatively correlated to α, L, and $\log(k_f/k_m)$. We also extended our results to infinitely-large reservoirs and presented regression-based models to estimate ρ_c and t from other characteristics.

Keywords: Effective permeability, Matrix-fracture systems, Numerical simulation, Percolation-based effective-medium approximation, Finite-size scaling theory

Table of Contents

List of Figures

List of Tables

Acknowledgements

Tolulope Agbaje is gratitude to the Department of Geology at Kansas State University for providing financial support in the form of an assistantship for the completion of this research. I would like to express my gratitude to AAPG for providing the AAPG Foundation scholarship. The financial assistances were instrumental in allowing me to focus on my studies and complete this research. I would also like to extend my deepest appreciation to my major advisor, Dr. Ghanbarian, for his guidance, encouragement, and unwavering support throughout the research process. I feel privileged to have had the opportunity to work with such an esteemed and knowledgeable advisor. In addition, I would like to thank my co-advisor, Dr. Jeffrey Hyman, for their invaluable contributions to this research. His expertise and support have been essential to the success of this project. I would also like to acknowledge all the support and help provided by the other faculty members and staff of the department during the course of this research. Finally, I would like to express my gratitude to my families and friends for their constant support and encouragement.

Dedication

I dedicate my master's degree to my beloved parents, who have always supported and encouraged me, my siblings, who have always been there for me and provided unwavering support, and my late Shola Ilesanmi, who played a significant role in my academic journey. Although he is no longer with us, his memory and influence will continue to inspire me.

Chapter 1 - Introduction

Accurate estimation of effective permeability (k_{eff}), the overall capability of a geologic formation to pass fluid through it, is essential for various processes, such as hydrocarbon exploration and production, aquifer management and remediation, hydrogen and carbon storage, and geothermal energy. However, modeling flow and transport in fractured geologic formations are challenging because they are complex and multi-scale systems typically composed of two components: (1) rock matrix, and (2) fractures, each of which contributes to reservoir properties, such as porosity and permeability. The k_{eff} value is, therefore, affected by both matrix (k_m) and fracture (k_f) permeabilities in fractured reservoirs. Depending on the density of fractures, the contribution of k_f to k_{eff} may be significant and orders of magnitude greater than that of k_m (Berre et al., 2019). Fracture networks have been extensively investigated in the literature (Nordqvist et al., 1992; Madadi et al., 2003; Bogdanov et al., 2003; Maillot et al., 2016; Viswanathan et al., 2018). For instance, the scale dependence of fracture networks has been well documented in the literature (Neuman 1994; de Dreuzy et al., 2002; Baghbanan & Jing, 2007; Azizmohammadi & Matthäi 2017; Forstner & Laubach, 2022). Vast evidence also indicates that k_{eff} depends on the fracture length power-law distribution and its exponent (e.g., de Dreuzy et al., 2001a; Berkowitz et al., 2000). The impact of fracture orientation on k_{eff} has been also investigated, with early work by Teufel et al. (1993) and recently by Zhu (2019). For a more recent review, see Viswanathan et al. (2022). Interconnected fractures within rock matrix tend to provide preferential pathways to fluid flow, and thus, dominantly control the overall permeability of such media. The k_{eff} value depends on geometrical and topological properties of fractures, such as aperture, width, length, orientation, and fracture density ρ (Ebigbo et al., 2016).

Various numerical methods were proposed and applied to study fluid flow and to determine the k_{eff} in fracture networks (Oda 1985; Long et al., 1985; Cacas et al., 1990; Durlofski 1991; Lough et al., 1997; Koudina et al., 1998; Nakashima et al., 2000; Jourde et al., 2002; Mustapha & Mustapha 2007). Among them, the discrete fracture network (DFN) approach was widely employed to model flow and transport (Dverstorp et al., 1992; Painter & Cvetkovic, 2005; Berrone et al., 2018). In the DFN method, governing equations of flow or transport are numerically solved in individual fractures. Although ingenious, this approach overlooks the permeability contribution from the rock matrix, assuming it to be negligible. To address this limitation, Sweeney et al. (2020) introduced the upscaled discrete fracture matrix (UDFM) model, an alternative to the traditional DFN approach, to more accurately capture interactions between fractures and surrounding rock matrix. Those authors demonstrated their model applicability to complex heterogeneous fractured media and validated it by comparing with numerical and analytical benchmarks.

In addition to numerical methods, theoretical models were developed to study k_{eff} in fracture networks and matrix-fracture systems. One of the early models is the power-law averaging (Journel et al., 1986; Deutsch, 1989). For instance, Zanon et al. (2002) applied power-law averaging to formations constructed of high-permeability sandstone and low-permeability shale grids. Those authors found that the power-law exponent (ω) depended on geological properties, such as the percentage of sandstone and anisotropy ratio. Years later, de Dreuzy et al. (2001b) applied the power-law averaging to permeability in two-dimensional fracture networks with apertures lognormally distributed. They reported that the exponent ω varied with network size, fracture length power-law distribution exponent, and fracture density. In another study, Mourzenko et al. (2011) proposed a heuristic model based on the asymptotic behavior of permeability in three-dimensional fracture networks, consistent with numerical data simulated

over a wide range of network densities. However, their model is not applicable to matrix-fracture systems where matrix contributes to fluid flow at very low fracture densities ($\rho < \rho_c$ where ρ and ρ_c are respectively fracture density and its critical value). More recently, following the work of Sævik et al. (2013), Ebigbo et al. (2016) evaluated various effective medium-based models, such as symmetric and asymmetric self-consistent, differential, and Maxwell, as well as the heuristic model of Mourzenko et al. (2011). They numerically simulated effective permeability in three-dimensional fractured rock masses composed of spheroidal fractures with different aspect ratios. In their simulations, Ebigbo et al. (2016) considered both mono- and poly-disperse fracture networks and varied the matrix permeability to fracture permeability ratio from 1.2×10^{-7} to 4.8×10^{-5}. They found good agreement between theoretical estimations and numerical simulations for the self-consistent effective-medium approximation. Ebigbo et al. (2016) also reported that the heuristic model of Mourzenko et al. (2011) was accurate, particularly for mono-disperse networks.

Percolation-based effective-medium approximation (P-EMA) proposed in physics literature by McLachlan (1987;1988) provides another theoretical framework to study fluid flow in two-component systems. However, to the best of our knowledge, neither has it been applied to matrix-fracture systems nor has it been attempted to model effective permeability in fractured reservoirs. Therefore, the main objectives of this study are to: (1) apply the P-EMA to fit k_{eff} as a function of ρ, (2) investigate how the P-EMA parameters (critical fracture density ρ_c and scaling exponent t) vary with fractures and reservoir properties, (3) study effects of heterogeneity on the k_{eff} in the matrix-fracture systems, and (4) address the effect of finite size on fractured reservoir and simulation of fluid flow through them.

Chapter 2 - Numerical Simulations

For the DFN simulations, we applied a parallelized computational platform called dfnWorks developed at the Los Alamos National Laboratory (Hyman et al., 2015). In contrast to the continuum model that represent the matrix-fracture system as a single continuous medium, the DFN method explicitly represents individual fractures in the network based on their attributes, such as length, orientation, and density. However, the DFN method does not account for the presence of rock matrix. To overcome this limitation, we utilized the Upscaled Discrete Fracture-Matrix (UDFM) model developed by Sweeney et al. (2020), which incorporates the effect of rock matrix and its contribution to the k_{eff}. In the following sections, we explain the fracture networks and flow simulations in further detail.

2.1. Fracture network generation

Within the dfnWorks computational platform, fracture networks are generated using the dfnGen package, which utilizes two different libraries: (1) feature rejection algorithm for meshing (FRAM) and (2) Los Alamos grid toolbox (LaGriT). Each DFN is constructed so that all features in the network, e.g., length of intersections between fractures and distance between lines of intersection of a fracture are greater than a user-defined minimum length scale (Hyman et al., 2015). Like most DFN modeling techniques, one of the key advantages of the feature rejection algorithm is its flexibility in accommodating any statistical survey of a fractured site for fracture network generation. This allows the generation of DFNs that are representative of naturally fractured sites. Observational data from fractured media in nature show that the probability density function of fracture lengths (l) is broad and typically conform to the following truncated power-law probability density function (Bonnet et al., 2001; de Dreuzy et al., 2012; Hyman et al., 2018):

$$f(l) = c_l l^{-\alpha}, \quad l_{min} \leq l \leq l_{max} \tag{1}$$

where l_{min} and l_{max} are the minimum and maximum fracture lengths, respectively, c_l is a constant coefficient (a normalization factor), and α is the exponent that controls the frequency of fracture length and, therefore, the heterogeneity of the network (de Dreuzy et al., 2012). The value of α varies typically between 1 and 3 (Bonnet et al., 2001).

Within the dfnWorks framework, each fracture has a specific orientation, which is sampled from the Fisher distribution

$$f(\beta, \kappa) = \frac{\kappa \sin(\beta) e^{\kappa \cos(\beta)}}{4\pi \sinh(\kappa)} \tag{2}$$

where β, which is the mean orientation vector, is the dip angle, κ is a concentration parameter controlling the uniformity and heterogeneity of the fracture orientation. The range of κ is wide and typically varies from 0 to ∞. κ values closer to zero tend to generate a more uniform distribution with all fracture orientations equally likely to be generated, while larger κ values tend to generate fracture orientation clustered around the mean orientation, ϕ and β.

Following Vermilye and Scholz (1995), we set l_{min} = 0.02 m and l_{max} = 20 m, in accord with the experimental observations reported for Florence Lake. Three different α values were considered (α = 1.5, 2.0, and 2.5) to cover the experimental range reported by Bonnet et al. (2001). We also assumed that fracture length and aperture were correlated. To further generate a polydisperse fracture network that matched the experimental data, we set κ = 0.1. As an example, we demonstrate several DFNs generated in this study in Fig. 1.

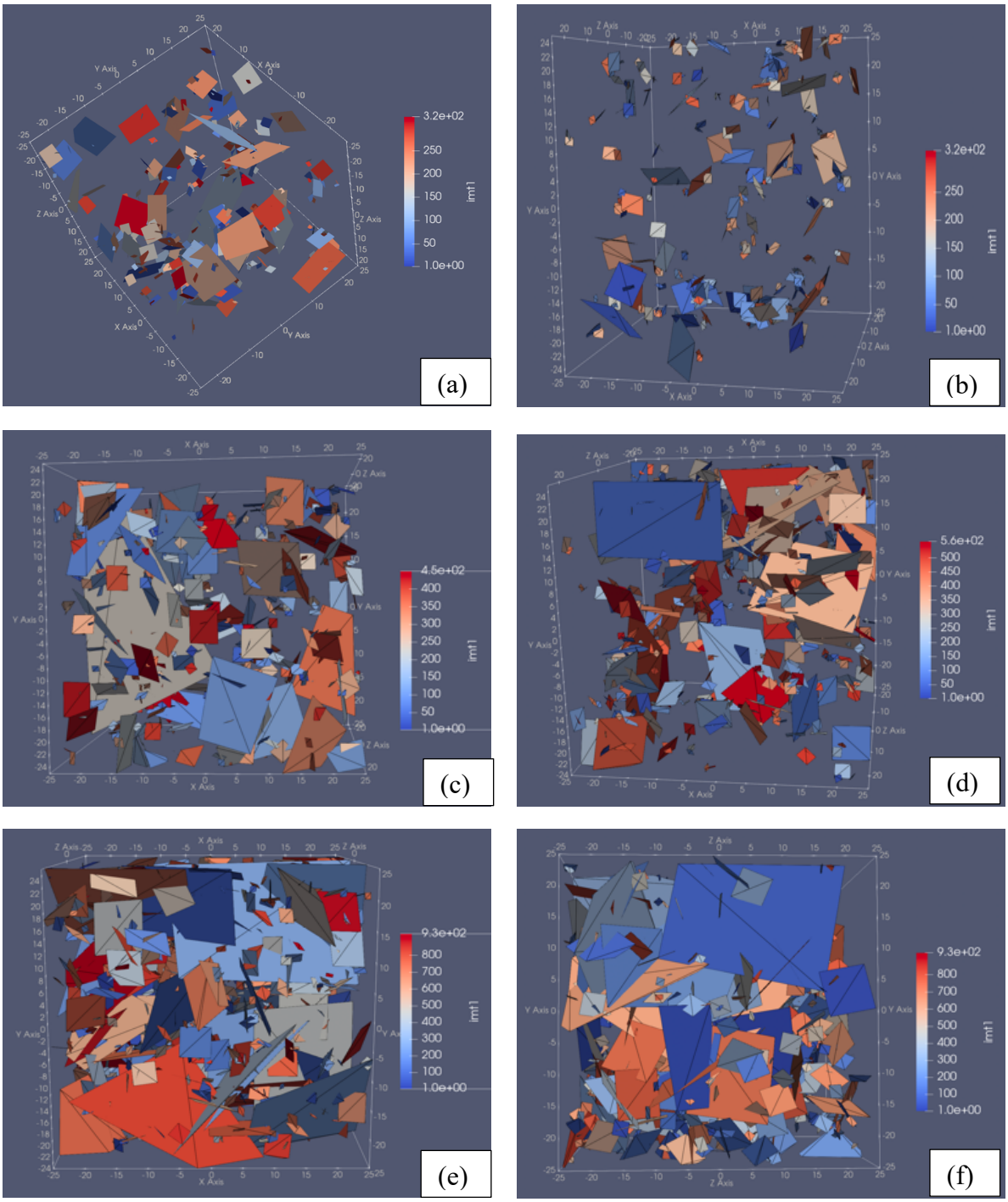

Figure 1. Stochastically generated DFNs with size L = 50 m using 2000 fractures. The left plots refer to log (k_f/k_m) = 2, while the right ones to log (k_f/k_m) = 6. α = 2.5 in top plots (a) and (b), 2 in middle plots (c) and (d), and 1.5 in bottom plots (e) and (f). Network generated at α =

1.5 appears to have more fractures, and specifically, longer fractures in the network, leading to early percolation relative to network generated at α = 2.0 and 2.5.

Although in all cases shown in Fig. 1 the number of fractures was 2000, the DFNs corresponding to α = 2.5 seem to be sparser compared to those corresponding to α = 1.5 and 2.0. This is because within the dfnWorks framework, isolated clusters are eliminated. As can be seen in Fig. 1, the networks generated with α = 1.5 tended to have longer fractures and better connected compared to the networks with α = 2 and 2.5. As Fig. 1 shows, the DFN generated with $\alpha = 2.5$ composed of shorter fractures, which ultimately resulted in relatively high percolation threshold because more fractures are needed to form a sample spanning cluster percolating the network.

To minimize uncertainties in the stochastically generated matrix-fracture systems and to obtain representative and statistically reliable results, we had multiple realizations. The number of realizations varied from one system to another depending on the value of α and the system size (L). We explain criteria used to determine the number of realizations and detail in the following section.

2.2. Meshing

We applied an octree-refined continuum mesh (Sweeney et al., 2020) to adequately capture the geometrical features of the generated matrix-fracture systems. The generated matrix-fracture systems were mapped onto a uniformly discretized hexahedral mesh to account for the presence of rock matrix which serve as the background of the mesh. After mapping the DFN onto the hexahedral mesh, the mesh was then binarized into fracture and matrix cells with cells intersected by fracture tagged “fracture cells” and cells not intersected by fracture tagged “matrix cells”. The

resolution of the mesh depends on the proximity to individual fractures within the fracture network. Higher mesh resolutions were employed near the fractures in the system which enabled us to capture the dynamic processes that occurs between the fractures and the surrounding matrix. The octree method utilizes two user-defined parameters: (1) edge length of the original hexahedral mesh before refinement (l) and (2) the number of refinement level (r) in the final octree mesh. In our simulation setup, l was set equal to $L/10$ and r equal to 2. Sweeney et al. (2020) conducted simulations using the UDFM method using three different values of r = 1, 2, and 3 and set l = L/10. Results from their simulations revealed that the value of r is less likely to significantly impact k_{eff}. However, they noted it was an important parameter to consider for solute transport problems. Therefore, our choice of r = 2 was to reduce the computational cost involved in meshing while ensuring more accuracy in k_{eff} calculation.

2.3. Fluid flow

Following the binarization of the system into fracture and matrix cells, their respective permeabilities (i.e., k_f and k_m) were upscaled into their respective cells. The upscaled properties were incorporated in the fluid flow simulation, and accordingly the value of k_{eff} was determined at different ρ values between 0 and 1. Fluid flow was simulated using PFLOTRAN, a parallel subsurface flow and reactive transport finite volume code (Lichtner et al., 2015). The PFLOTRAN employs the Richards' equation, extensively applied in subsurface hydrology for modeling flow under a single phase, variably saturated, isothermal, and steady stay conditions, to compute the outlet volumetric flow rate. Pressure boundary conditions at the inlet and outlet was set at 1.1×10^6 and 1.0×10^6 Pa, respectively. We then numerically invert the Darcy's equation (Eq. (3)) to calculate the k_{eff} of individual DFN in the network.

$$q = -k_{eff}.\Delta p/L \quad (3)$$

$$k_{eff} = -q.L/\Delta p \quad (4)$$

where q is the Darcy's flux, calculated by dividing the volumetric flow rate Q, which is computed by PFLOTRAN, by the area of the system ($L \times L$), Δp is the change in pressure from the inflow to outflow boundary, and L is the system size.

In this study, the k_m was set equal to $10^{-18}\ m^2$ in accord with shale matrix permeability values reported by Best and Katsube (1995) and Wang et al. (2009). Three fracture permeability values k_f = 10^{-12}, 10^{-14}, and $10^{-16}\ m^2$ corresponding to log (k_f/k_m) = 2, 4, and 6 were considered. These values, k_m and k_f, were upscaled in their appropriate cells, i.e., matrix or fracture cell.

For a given complete simulation, we extracted the value of k_{eff} and computed the corresponding fracture density (ρ), i.e., the fraction of volume occupied by fracture within the matrix-fracture system to the overall volume of the system, which were subjected to further analyses. The equation for fracture density is given by:

$$\rho = v_f/v \quad (5)$$

Where v_f is the volume occupied by fracture in the system while v is the total volume of the system.

As stated earlier, we conducted iterations of the matrix-fracture system generation and fluid flow simulations to obtain statistically reliable results. We started with 10 iterations at each ρ, and then computed average, variance and 95% confidence intervals (CIs) of the k_{eff}. The mean k_{eff} was then plotted against the corresponding mean ρ including the 95% CIs. We added individual iteration to the plot, eliminated any iteration that falls outside the 95% CIs from further analyses, and calculated the standard deviation of the iterations that fall within the 95% CIs. More iterations

were conducted, and the above process is repeated until the standard deviation no longer changed with the number of realizations.

The total number of iterations required to achieve statistically reliable results for each matrix-fracture system realization is dependent on the level of heterogeneity, which is largely controlled by α and L. For α values of 2.5 and 2.0, a minimum of 10 iterations were conducted for each ρ in a complete set of simulations, comprising 6 data points. However, due to the increased heterogeneity near the critical fracture density, which is the minimum ρ required for percolation to occur within the system, a minimum of 20 iterations were conducted at intermediate fracture densities where the transition of k_m to k_f happens. Consequently, a single matrix-fracture system realization generated at α values of 2.5 and 2.0 and $L \geq 50\ m$ had a total of at least 80 iterations. For smaller values of L, where we observed an increased heterogeneity, such as L = 22.5 and 30 m, we conducted at least 20 iterations for each data point irrespective of the value of α. The DFNs generated with $\alpha = 1.5$, as discussed earlier, has a higher tendency to generate longer fracture in the network, which significantly increased the level of heterogeneity in the matrix-fracture system relative to the DFNs generated with α = 2.5 and 2.0. Therefore, for $\alpha = 1.5$, we had at least 20 iterations for each ρ value in a complete set of simulation, and a minimum of 30 iterations were conducted around the critical fracture density. In total, we conducted over 5000 simulations for different matrix-fracture systems studied here and the summary is detailed in Table 1 below.

Table 1. Summary of simulation inputs and the corresponding number of simulations generated.

α	$\log(k_f/k_m)$	L (m)	Total number of ρ (range)	Average number of iterations for each ρ	Total number of runs
1.5	2, 4, 6	22.5, 30, 50, and 70	6 (0 – 1)	30	2160
2.0	2, 4, 6	22.5, 30, 50, and 70	6 (0 – 1)	20	1440
2.5	2, 4, 6	22.5, 30, 50, and 70	6 (0 – 1)	20	1440

Chapter 3 - Theory

Percolation-based effective-medium approximation (P-EMA) is an upscaling technique from statistical physics originally proposed by McLachlan (1987;1988) for media with low- and high-conductivity components. It includes percolation theory and effective-medium approximation as its special cases (Ghanbarian and Daigle, 2016). Although previously applied to study electrical conductivity, thermal conductivity, permeability and Young's modulus in binary composites (Deprez et al., 1988; McLachlan, 2021), its applications to porous rocks and reservoirs have been very limited. To the best of our knowledge, the P-EMA has not yet been applied to model the k_{eff} in matrix-fracture systems.

Within the P-EMA framework, the relationship between k_{eff} and ρ is given by

$$(1-\rho)\frac{k_m^{\frac{1}{t}}-k_{eff}^{\frac{1}{t}}}{k_m^{\frac{1}{t}}+\left[\frac{1-\rho_c}{\rho_c}\right]k_{eff}^{\frac{1}{t}}}+\rho\frac{k_f^{\frac{1}{t}}-k_{eff}^{\frac{1}{t}}}{k_f^{\frac{1}{t}}+\left[\frac{1-\rho_c}{\rho_c}\right]k_{eff}^{\frac{1}{t}}}=0 \qquad (6)$$

where ρ_c is the critical fracture density and t is the scaling exponent. In Eq. (6), k_{eff} is implicitly explained in terms of ρ. Rearranging Eq. (6) gives ρ explicitly as a function of k_{eff} as follows

$$\rho=\frac{\left(k_{eff}^{\frac{1}{t}}-k_m^{\frac{1}{t}}\right)\left(\rho_c k_f^{\frac{1}{t}}+(1-\rho_c)k_{eff}^{\frac{1}{t}}\right)}{\left(k_f^{\frac{1}{t}}-k_m^{\frac{1}{t}}\right)k_{eff}^{\frac{1}{t}}} \qquad (7)$$

Note that for $\rho < \rho_c$, k_{eff} is mainly controlled by the rock matrix, while for $\rho > \rho_c$, k_{eff} is dominated by the fracture network and its properties. In this study, we fit Eq. (7) to the averaged $\rho - k_{eff}$ curves using the nonlinear least square optimization method, optimized the P-EMA model parameters i.e., ρ_c and t, and explored the relationship between the matrix-fracture system properties and the P-EMA model parameters.

Chapter 4 - Results and Discussion

In this section, we present the results of k_{eff} as a function of ρ for different matrix-fracture systems, fitting the P-EMA model to the k_{eff} – ρ simulations for 36 matrix-fracture systems, relationship between the P-EMA model parameters and matrix-fracture system properties, and the extension of our results to an infinitely-large fractured reservoirs.

4.1. Effect of α on k_{eff} in matrix-fracture systems

Fig. 2 shows the behavior of k_{eff} as a function of ρ for $\alpha = 2.5$, $\log(k_f/k_m) = 6$, and $L = 50$ m. As observed, at lower ρ values, k_{eff} remains nearly constant and the hydraulic properties of the matrix-fracture system are controlled by the rock matrix. At some intermediate fracture density, however, fluid finds a conductive pathway through the fracture network as a result of fracture connectivity, which results in a significant increase in k_{eff}. At higher ρ values, the increase in k_{eff} become stable and the $k_{eff} - \rho$ curve become flattened toward k_f yielding a sigmoidal shape (Fig. 2). This intermediate fracture density corresponds to the percolation threshold; therefore, we identified two distinct regions: (1) the matrix-controlled region which is below the percolation threshold and (2) the fracture-controlled region which is above the percolation threshold.

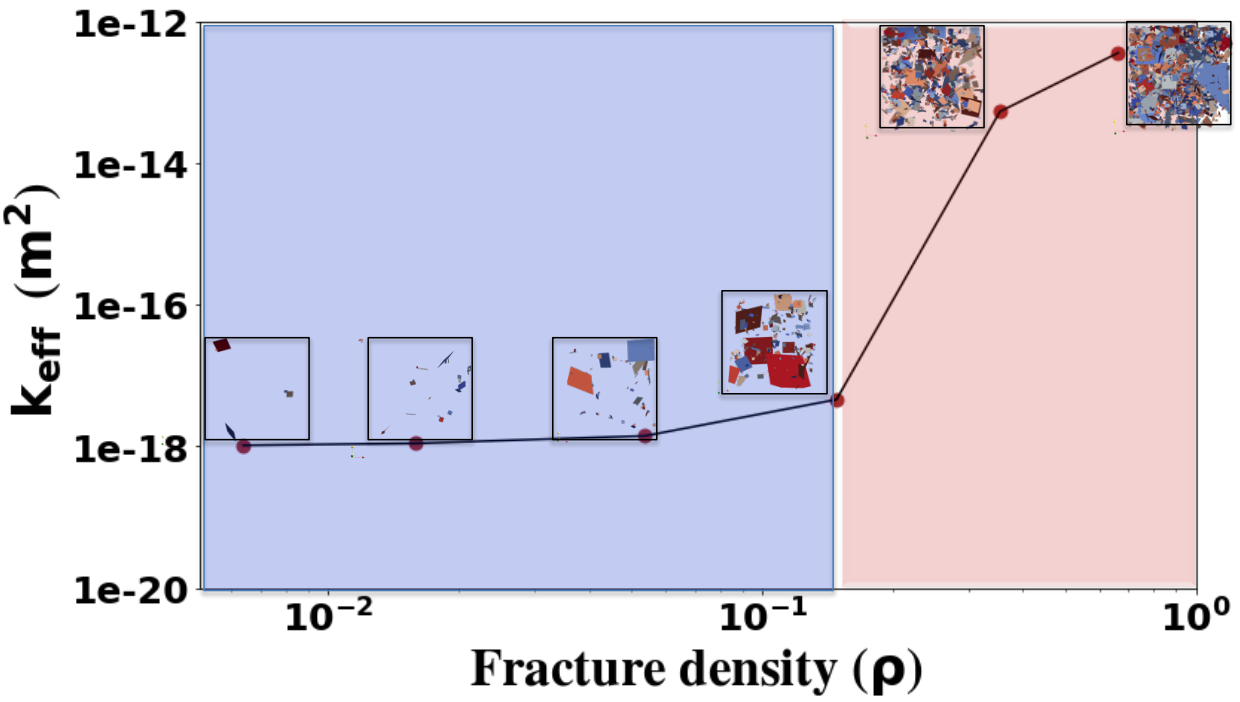

Figure 2. Effective permeability k_{eff} as a function of fracture density ρ alongside fracture network generated at each ρ value. The region colored in blue, where we have sparse fracture networks, is regarded as the matrix-controlled region, and the region in red, where we have denser fracture networks, is the fracture-controlled region.

Fig. 3 shows that for α = 1.5 the transition from the matrix-controlled region to the fracture-dominated one occurred at a lower fracture density ρ compared to α = 2.5. Additionally, as observed in Fig. 3, the transition from matrix permeability to fracture permeability becomes clearer and more significant as the $\log(k_f/k_m)$ value becomes greater.

Results presented in Fig. 1 confirm that lower α values resulted in longer fractures within the fracture network, which significantly impacted connectivity and consequently the k_{eff}. This is consistent with the results of Berkowitz et al. (2000) and others who reported that the connectivity of fracture networks was dependent on the exponent α. Accordingly, we expect the lower alpha values to have early percolation of the fracture network relatively to fracture network generated at higher α values.

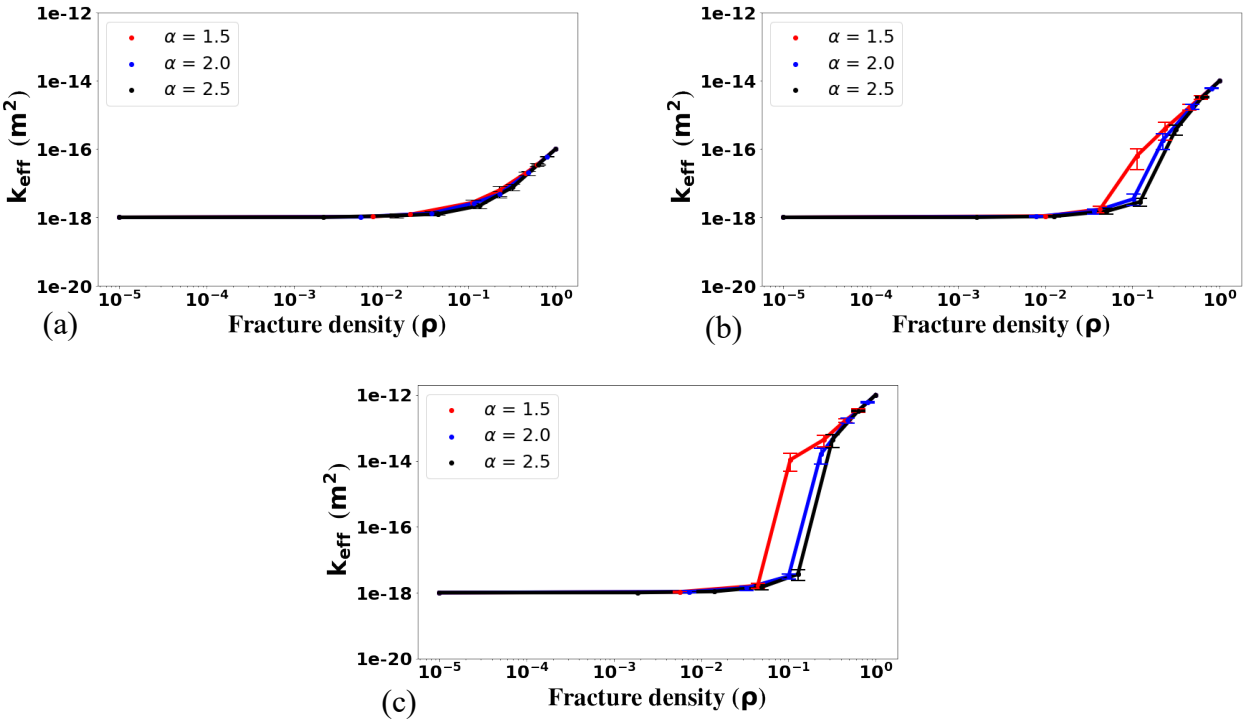

Figure 3. Effective permeability, k_{eff}, versus fracture density, ρ, for (a) log (k_f/k_m) = 2, (b) log (k_f/k_m) = 4, and (c) log (k_f/k_m) = 6 and different α values. In all cases, the system size, L = 50 m and the matrix permeability $k_m = 10^{-18}\ m^2$. Each data point represents the average over multiple iterations and the error bars correspond to one standard deviation. The impact of α appears to be more significant with increasing log (k_f/k_m).

Zhu et al. (2021) investigated how geometrical properties of fracture networks, such as length, orientation, aperture, and position of fracture centers, affect macro-scale flow properties in shale-like formations. They represented the fracture network as a graph where individual fracture is a node, and used global efficiency, which is the average inverse shortest path length between all pairs of nodes in the graph, to measure the network's connectivity. They found that as the fracture length power-law distribution exponent (α) increased, the global efficiency of the network

decreased, i.e., the connectivity of the system decreased with increasing values of α, and more fractures were required to achieve percolation. Additionally, they reported that flow rate increased as α decreased, which is consistent with our results shown in Fig. 3.

4.2. Effect of $\log(k_f/k_m)$ on k_{eff} in matrix-fracture systems

In Fig. 4, we show the value of k_{eff}, averaged over several realizations, as a function of ρ for $\log(k_f/k_m)$ = 2, 4, and 6 and α = 1.5, 2, and 2.5. Our results clearly indicate that the percolation threshold became more significant with increasing value of $\log(k_f/k_m)$, as illustrated in Fig. 4. This means that in a connected fractured media, the magnitude at which fracture controls flow is dependent on the ratio of fracture to matrix permeability, $\log(k_f/k_m)$. Comparing Figs. 4a, 4b, and 4c also shows that such a transition happened at a larger fracture density as the value of α increased from 1.5 (Fig. 3a) to 2.5 (Fig. 3c).

Our results on percolation threshold being more significant as the value of $\log(k_f/k_m)$ increases agree with the results of Hyman et al. (2018) who reported that the magnitude of effective permeability increase around the percolation threshold was greater in matrix-fracture systems of greater $\log(k_f/k_m)$. Results presented in Fig. 4 are also consistent with those reported by Ebigbo et al. (2016) who found that in systems with small perturbations in matrix and fracture permeability values, the simulated $k_{eff} - \rho$ data exhibited nearly a linear trend with smooth transition from matrix to fracture permeability. However, the percolation threshold was more distinctive as the perturbation increased. In their study, the ratio of matrix to fracture permeability was observed to impact the effect of α and L on percolation within the matrix-fracture systems. Finally, in agreement with Zhu et al. (2021) where they reported that flow rate increases with increasing ratio

of matrix and fracture permeability, at $\rho > \rho_c$, k_{eff} was observed to increase with increasing values of $\log(k_f/k_m)$.

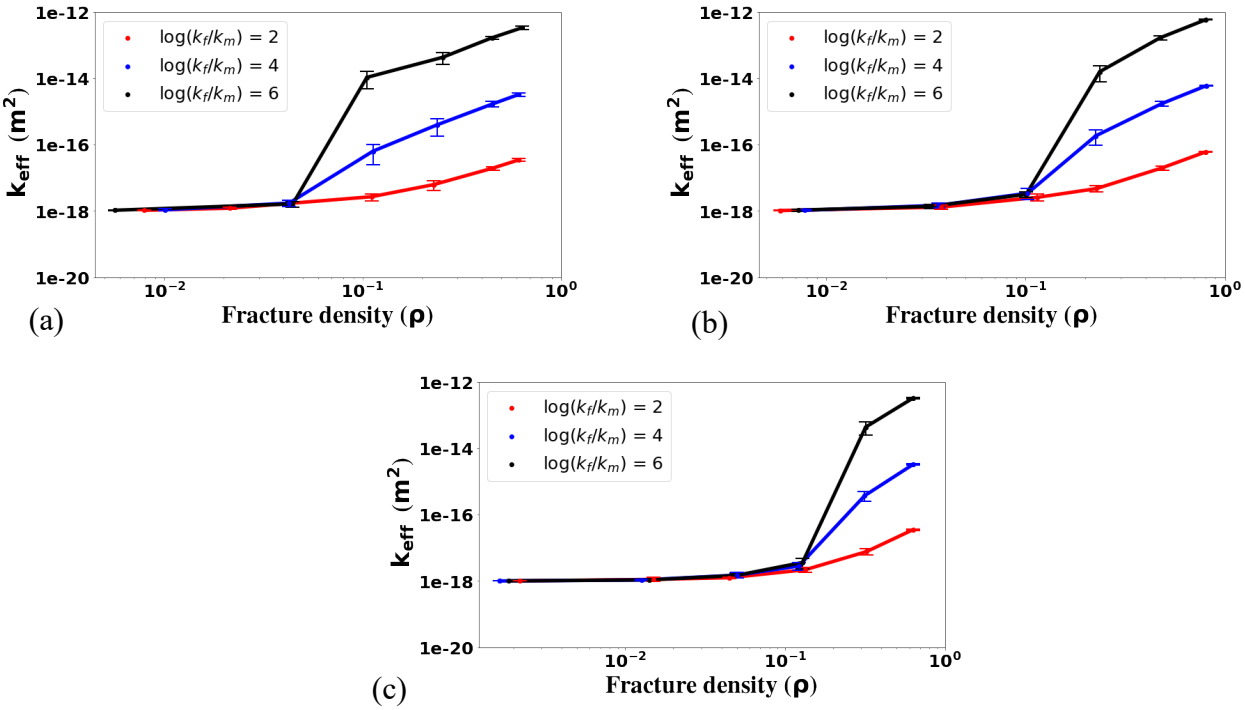

Figure 4. Effective permeability, k_{eff}, versus fracture density, ρ, for (a) $\alpha = 1.5$, (b) $\alpha = 2.0$, and (c) $\alpha = 2.5$ and different log (k_f/k_m) values. In all cases, the system size, L = 50 m and the matrix permeability $k_m = 10^{-18}\ m^2$. Each data point represents the average over multiple iterations and the error bars correspond to one standard deviation.

4.3. Effect of scale L on k_{eff} in matrix-fracture systems

Understanding the scale dependence of intrinsic properties of porous media such as permeability is important for making inference about behavior of such properties at the global scale from local scale simulations. We investigated the scale dependence of k_{eff} in matrix-fracture systems by plotting k_{eff} as a function of ρ for different L values (Fig. 5). For a given simulation with

consistent input parameters at different values of L, at $\rho < \rho_c$ we observed that k_{eff} remained almost the same for L = 22.5, 30, 50, and 70 m. At $\rho > \rho_c$, we found that lower L values exhibited higher k_{eff} and k_{eff} tended to decrease as L increased, which is consistent with the findings of Lei et al. (2015). de Dreuzy at al. (2001a) also reported similar behavior for DFNs generated at $\alpha < 3$, consistent with the range of α considered in our simulations.

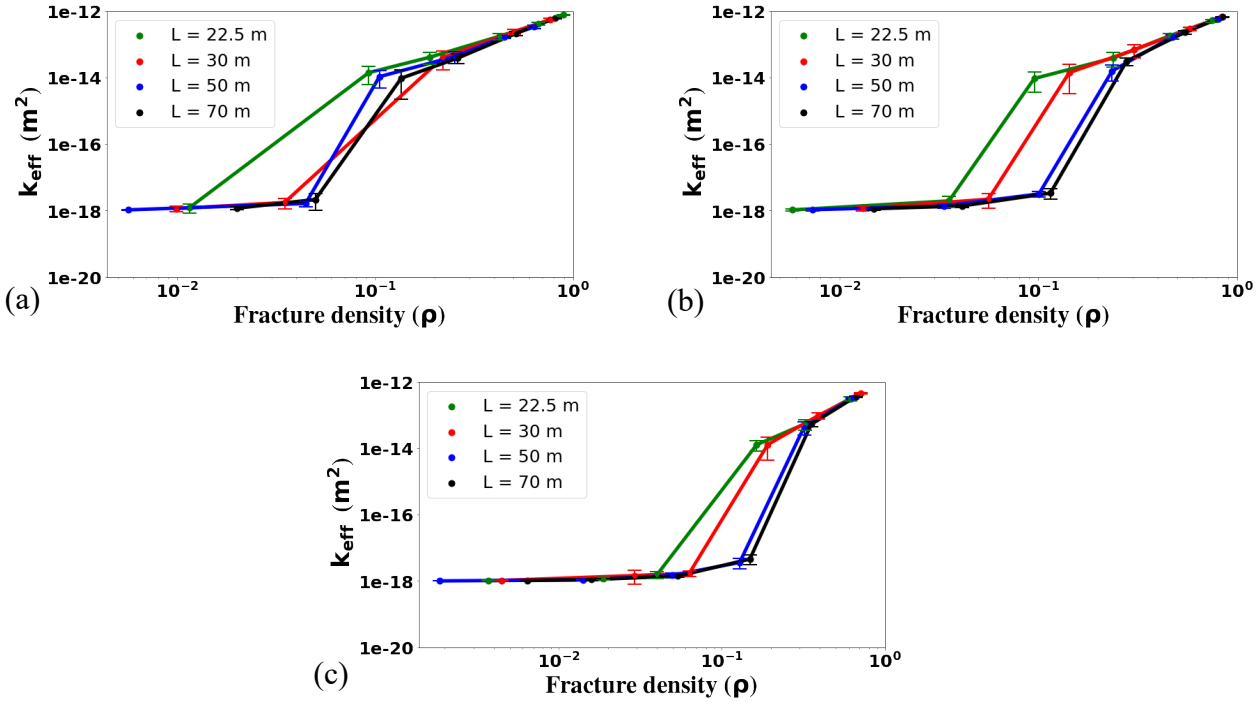

Figure 5. Effective permeability, k_{eff}, versus fracture density, ρ, for (a) $\alpha = 1.5$, (b) $\alpha = 2.0$, and (c) $\alpha = 2.5$ and different L values. In all cases, $\log(k_f/k_m) = 6$ and the matrix permeability $k_m = 10^{-18}\ m^2$. The critical fracture density tends to increase with increasing L. Each data point represents the average over multiple iterations and the error bars correspond to one standard deviation.

As observed from Fig. 5, percolation occurred relatively early at low L values, and the percolation threshold tends to increase with increasing L, in accord with the finite-size scaling analysis (Stauffer and Aharony, 2018) as we discuss later. This observation is consistent for the networks generated at α values of 1.5, 2.0, and 2.5. However, for α = 1.5 (Fig. 5a), the percolation threshold for L = 30, 50, and 70 m were observed to be very close to one another. A plausible explanation for this finding could be as a result of networks generated with α value of 1.5 tend to have higher frequency of longer fractures as exemplified by Fig. 1, which results in early percolation of the network irrespective of the system size. Additionally, it was also noted that at α value of 1.5, networks generated at L = 22.5 m were observed to have a much earlier percolation relative to networks generated at L = 30, 50, and 70m. We suggested that this observation could be due to the fact that the system size, L = 22.5 m is closer to the l_{max} defined in our simulation setup, which is 20 m. This means that for a matrix-fracture system generated at α = 1.5, where the L is closer to the l_{max}, there is a higher probability that a single fracture can propagate the system, causing a much earlier percolation of the system.

4.4. Percolation-based effective-medium approximation

Using the Curve Fitting Toolbox of MATLAB, the P-EMA, Eq. (7), was fit to the $\rho - k_{eff}$ simulations averaged over various realizations. The optimized values of the P-EMA parameters, ρ_c and t, for the different matrix-fracture systems studied here are summarized in Table 2. As an example, the P-EMA fits for L = 50 m and different values of $\log(k_f/k_m)$ and α are shown in Fig. 6.

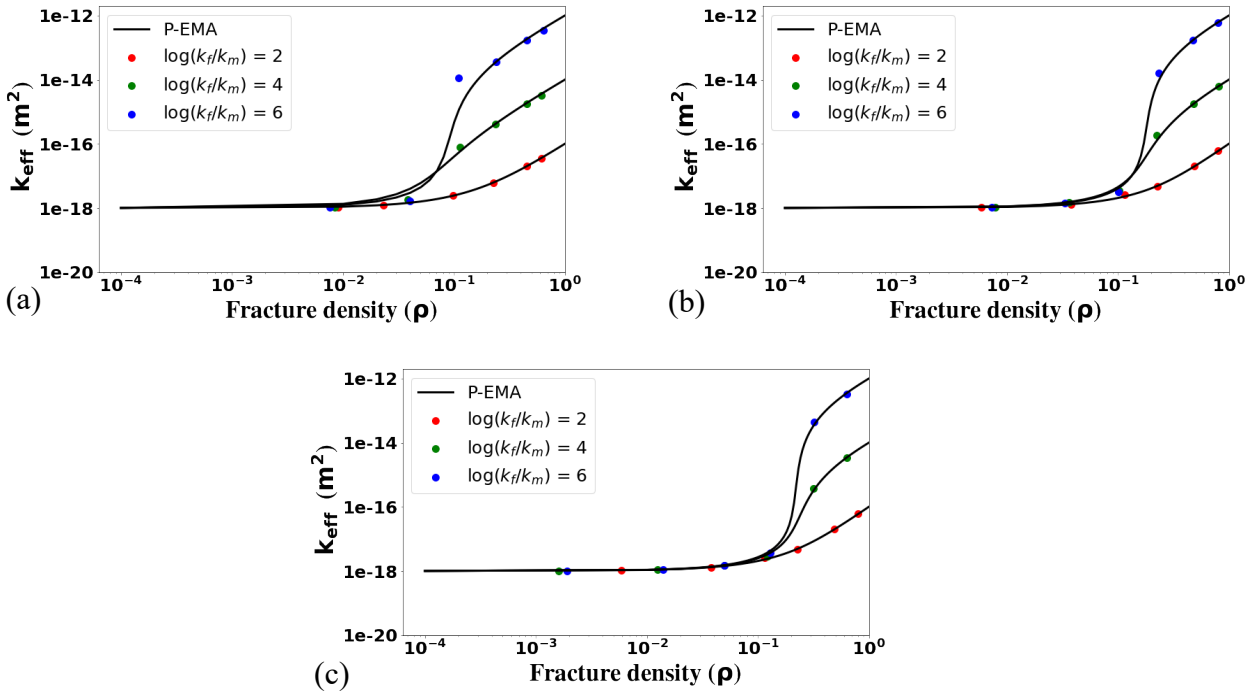

Figure 6. The fit of the P-EMA to the $k_{eff} - \rho$ simulations for (a) $\alpha = 1.5$, (b) $\alpha = 2.0$, and (c) $\alpha = 2.5$ and different log (k_f/k_m) values. In all cases the system size L = 50 m. The optimized parameters of the P-EMA are summarized in Table 1.

The high values of R^2 (= 0.99) reported in Table 2 imply that the P-EMA (Eq. 7) fit the data well. We observed that the smaller the α value the lower the ρ_c, meaning that the lower α values resulted in early percolation of fracture network. We found the lowest value of ρ_c in the matrix-fracture systems with α = 1.5, followed by α = 2.0, while the matrix-fracture systems with α = 2.5 had the highest value of ρ_c. Our results, however, are not consistent with those of de Dreuzy et al. (2000) who reported that by increasing α the percolation threshold in three-dimensional fracture networks decreased. de Dreuzy et al. (2000) stated that such a trend may be attributed to truncation effect because in their study elliptical fractures truncated by the sides of system had an internal

characteristic length less than the original one. In addition to that, in their study α ranged between 2.5 and 5, while in our study between 1.5 and 2.5 in accord with the experimental range reported for naturally fractured site (Bonnet et al., 2001). Recall that α controls the frequency of fracture length; the lower the α value, the greater the number of longer fractures (Fig. 1). Therefore, the correlation between ρ_c and α seems reasonable.

Scaling exponent, t, which is the other parameter of the P-EMA model, was observed to decrease as the value of α increased. For example, the average value of t for all matrix-fracture system realizations considered when α = 1.5 is 2.07, but decreases to 2.02 and 1.85 for α = 2.0 and α = 2.5 respectively. Additionally, increasing the system size, L, was also observed to result in a decrease in value of t. For L values of 22.5 m, 30 m, 50 m, and 70 m, the corresponding average values of t for all matrix-fracture system realizations considered are 2.15, 2.05, 1.87, and 1.84 respectively. Moreover, our analyses revealed that t exhibits an inverse relationship with $\log(k_f/k_m)$. The average values of t for $\log(k_f/k_m)$ = 2, 4, and 6 are 2.19, 1.92, and 1.83 respectively for all matrix-fracture system realizations considered. Since $\log(k_f/k_m)$ controls the shape of the $k_{eff} - \rho$ curve and it is inversely correlated to t, one may expect that the lower value of the scaling exponent t results in sharper increase in the k_{eff} corresponding to larger contrast in matrix and fracture permeabilities.

Table 2. The optimized values of the P-EMA model parameters for different matrix-fracture systems generated in this study. k_m = 10^{-18} m^2 consistent with shale matrix permeability.

Reservoir	α	*System size L (m)*	$\log(k_f/k_m)$	ρ_c	t	R^2
1	2.5	22.5	2	0.13	2.33	0.99
2	2.5	22.5	4	0.11	2.07	0.99
3	2.5	22.5	6	0.12	1.98	0.99
4	2.0	22.5	2	0.09	2.39	0.99
5	2.0	22.5	4	0.07	2.15	0.99
6	2.0	22.5	6	0.07	2.07	0.99
7	1.5	22.5	2	0.00	2.43	0.99
8	1.5	22.5	4	0.00	2.05	0.99
9	1.5	22.5	6	0.00	1.99	0.99
10	2.5	30	2	0.16	2.25	0.99
11	2.5	30	4	0.12	1.97	0.99
12	2.5	30	6	0.12	1.90	0.99
13	2.0	30	2	0.09	2.36	0.99
14	2.0	30	4	0.07	2.04	0.99
15	2.0	30	6	0.06	1.99	0.99
16	1.5	30	2	0.06	2.48	0.99
17	1.5	30	4	0.06	2.04	0.99
18	1.5	30	6	0.08	1.90	0.99
19	2.5	50	2	0.26	1.81	0.99
20	2.5	50	4	0.23	1.65	0.99
21	2.5	50	6	0.22	1.63	0.99
22	2.0	50	2	0.17	2.32	0.99
23	2.0	50	4	0.17	1.81	0.99
24	2.0	50	6	0.18	1.73	0.99
25	1.5	50	2	0.06	2.50	0.99
26	1.5	50	4	0.05	2.09	0.99
27	1.5	50	6	0.06	2.03	0.99
28	2.5	70	2	0.28	1.77	0.99
29	2.5	70	4	0.24	1.64	0.99
30	2.5	70	6	0.24	1.63	0.99
31	2.0	70	2	0.22	2.03	0.99
32	2.0	70	4	0.19	1.77	0.99
33	2.0	70	6	0.19	1.69	0.99
34	1.5	70	2	0.13	2.33	0.99
35	1.5	70	4	0.07	2.12	0.99
36	1.5	70	6	0.06	2.07	0.99

4.5. Relationships between P-EMA parameters and matrix-fracture system properties

In this section, we further investigated the relationship between the P-EMA model parameters (ρ_c and t) and other matrix-fracture system properties through stepwise multiple-linear regression analysis. We should emphasize that our aim here is to better understand statistically which properties control variation in ρ_c and t values.

To establish multiple-linear regression models, two dependent variables, ρ_c and t, and several independent variables, i.e., α, L, log (k_f/k_m), l_{min}/L, l_{max}/L, k_m/L, $\sqrt{k_m}/L$, $\sqrt{k_m}/l_{min}$, and $\sqrt{k_m}/l_{max}$ were used. We found

$$\rho_c = 0.002 + 0.122\alpha - 0.204(l_{max}/L),\ R^2 = 0.94 \quad (8)$$

$$t = 3.044 - 0.218\alpha - 0.089\left(\log\left(k_f/k_m\right)\right) - 0.006L,\ R^2 = 0.83 \quad (9)$$

Regression-based results showed that ρ_c was significantly and statistically linked to α and L (p-value of < 0.0001) consistent with our previous results stated earlier. Although positive correlation between ρ_c and α was also reported by Mourzenko et al. (2005) for fracture networks with $\alpha < 3$, Sahimi and Mukhopadhyay (1996) found an inverse relationship. They, however, studied the scale dependence of percolation threshold in networks with long-range correlations. Drawing upon our previous analysis of the impact of L on k_{eff}, we found that when holding all other input parameters constant, an increase in L leads to an observed increase in the percolation threshold within the system. This implies that the creation of more fractures is required in larger system configurations in order to achieve percolation relative to smaller L. This supports the results obtained from our regression analysis, which confirms that L has a significant influence of ρ_c. Specifically, the observed increase in the percolation threshold with increasing L can be attributed to the fact that

larger L results in a longer possible path for connectivity to occur, thus necessitating the generation of a larger number of fractures for the system to percolate. In contrast, for smaller values of L, the system is simpler and has a shorter possible path for percolation to occur, which means that relatively fewer fractures are required for the system to percolate.

$\log(k_f/k_m)$, however, was observed to have no statistically significant relationship with ρ_c, as revealed by a p-value of 0.09, which exceeds the defined level of significance of 0.05. This outcome is reasonable given permeability ratio of fracture and rock matrix is a hydraulic property of the system and not a geometric property. Geometric properties of the system are only anticipated to impact ρ_c.

Our regression-based results showed that the independent variables α, $\log(k_f/k_m)$, and L statistically and significantly contributed to the dependent variable t (p-value < 0.0001). We also found that t was negatively correlated to α, L, and $\log(k_f/k_m)$ (see Eq. (9)). such dependencies suggest that t is influenced by geometrical properties of the matrix-fracture system as well as the hydraulic properties of the system.

-Predicting P-EMA parameters

We applied the developed regression-based model, i.e., Eq. (8) and Eq. (9) to predict ρ_c and t respectively. We conducted new set of simulations based on outcrop data of a naturally fractured site in order to evaluate the model performance on different input data. The summary of the input parameters for the simulations alongside the corresponding predicted values of ρ_c and t is presented in Table 3.

Table 3. Summary of simulation inputs and the corresponding predicted values of ρ_c and t.

Reservoir	$\boldsymbol{\alpha}$	$\boldsymbol{\log(k_f/k_m)}$	$\boldsymbol{L\ (m)}$	$\boldsymbol{l_{min}\ (m)}$	$\boldsymbol{l_{max}\ (m)}$	**Predicted** $\boldsymbol{\rho_c}$	**Predicted** $\boldsymbol{t}$
1	1.75	5	100	0.5	13	0.19	1.51
2	1.75	7	100	0.5	13	0.19	1.33
3	2.25	5	100	0.5	13	0.25	1.62
4	2.25	7	100	0.5	13	0.25	1.44
5	2.83	9	18	1.1	10	0.23	1.52
6	2.37	9	6	0.3	3	0.19	1.69

Data from the first four rows were obtained from the Culpeper quarry fractured site (Vermiyle and Scholz, 1995) and data from the fifth and sixth rows were obtained from the Hornelen1 bed (Azizmohammadi and Matthäi, 2017) and Kilve bed (Lei et al., 2015) of the Bristol channel basin. The predicted values of ρ_c and t were then inserted in Eq. (7) and used to predict the k_{eff} as a function of ρ. Fig. 7 shows the plot of k_{eff} as a function of ρ for the new set of simulations as well as the prediction. As observed from Fig. 7, the predictions closely match the simulated datapoint with an average R^2 of 0.98 and absolute error ranging from 16.45 % to 75.26 % which suggests a good performance of the model.

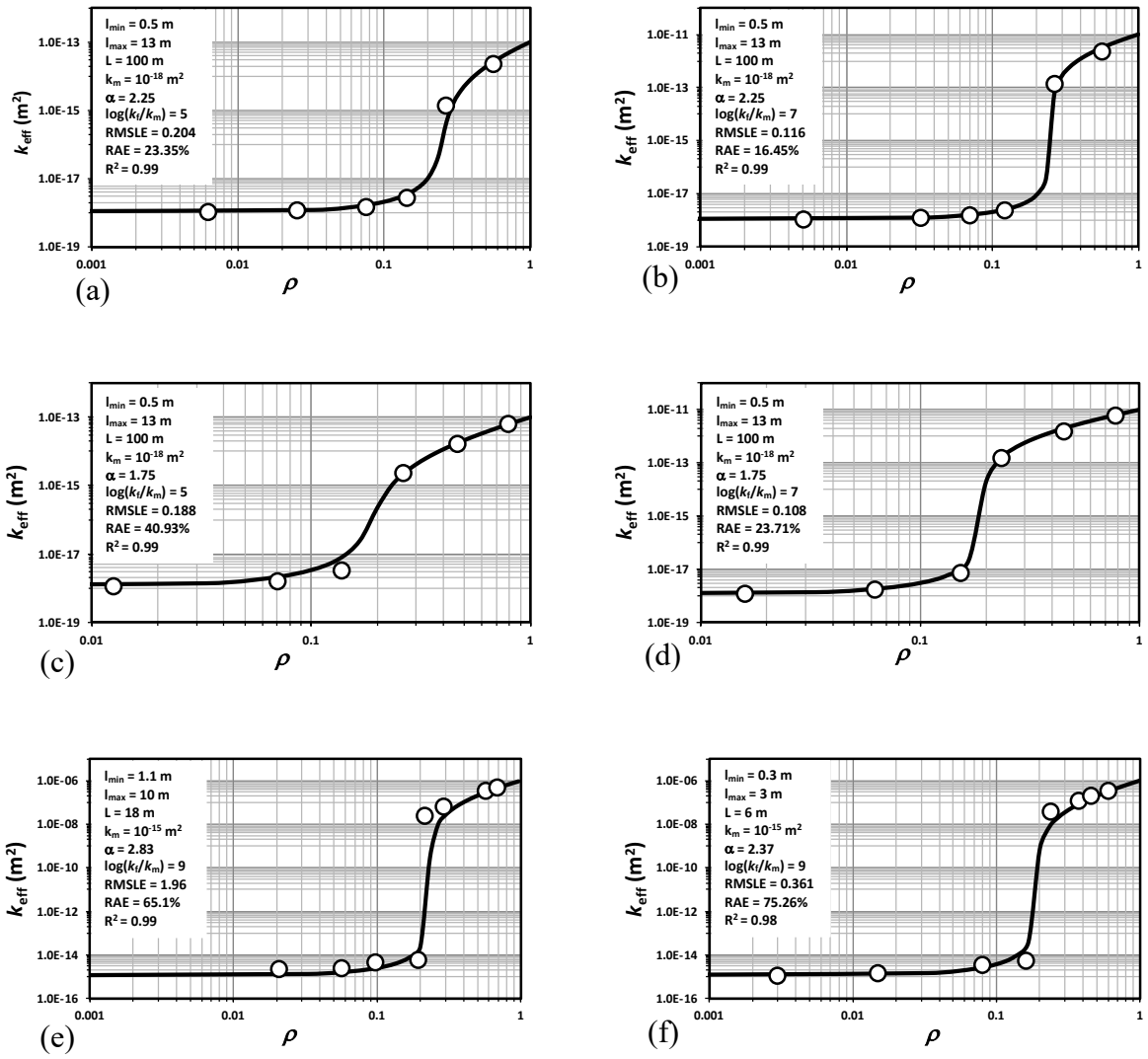

Figure 7. Effective permeability k_{eff} as a function of fracture density ρ. Each plot shows the simulated data in circle and prediction in solid black line.

4.6. Extrapolation to infinitely-large fractured reservoirs

Modeling fluid flow within the actual size of a matrix-fracture system as observed in nature, which may be considered as an infinitely-large medium, would be challenging due to computational costs. In this section, we apply concepts of finite-size scaling analysis (Stauffer and Aharony, 2018), to approximate the value of ρ_c in fractured reservoirs whose dimensions are infinitely large.

We also use an empirical exponential relationship proposed by Matyka et al. (2008) to estimate the value of t in large matrix-fracture systems.

- ρ_c for infinitely large systems

Within the finite-size scaling analysis framework, one can explain the scale dependence of ρ_c as follows (Stauffer & Aharony, 2018):

$$\rho_c(L) - \rho_c(L \to \infty) = CL^{-\frac{1}{\nu}} \quad (10)$$

where $\rho_c(L)$ and $\rho_c(L \to \infty)$ are the critical fracture densities for a finite- and infinite-sized matrix-fracture systems, respectively, C is a constant coefficient whose units depends on the system units, and ν is the correlation length exponent equal to 0.88 in three dimensions (Hunt et al., 2014). Finite-size scaling analysis has been widely applied in the literature (Sahimi, 2011). For example, Ji et al. (2004) let C and ν to be fitting parameters, fit Eq. (10) to simulations on two-dimensional fracture networks and reported C = 0.15 and ν = 1.27. The latter is slightly less than the universal ν value in two dimensions i.e., 1.33 (Hunt et al., 2014). Mourzenko et al. (2011) also applied a model similar to Eq. (10) to extrapolate permeability to infinitely large networks.

Fig. 8 shows ρ_c as a function of $L^{-\frac{1}{\nu}}$ and the fit of Eq. (10) to the corresponding data. As can be seen, Eq. (10) fit the data well with an average R^2 = 0.98. We found that the optimized value of C, reported in Fig. 8, decreased with increasing log (k_f/k_m). For instance, for $\log(k_f/k_m)$ = 2, 4, and 6, C was respectively 6.93, 6.58, and 6.5 when α = 1.5, while 4.60, 3.68, and 3.20 when α = 2.5.

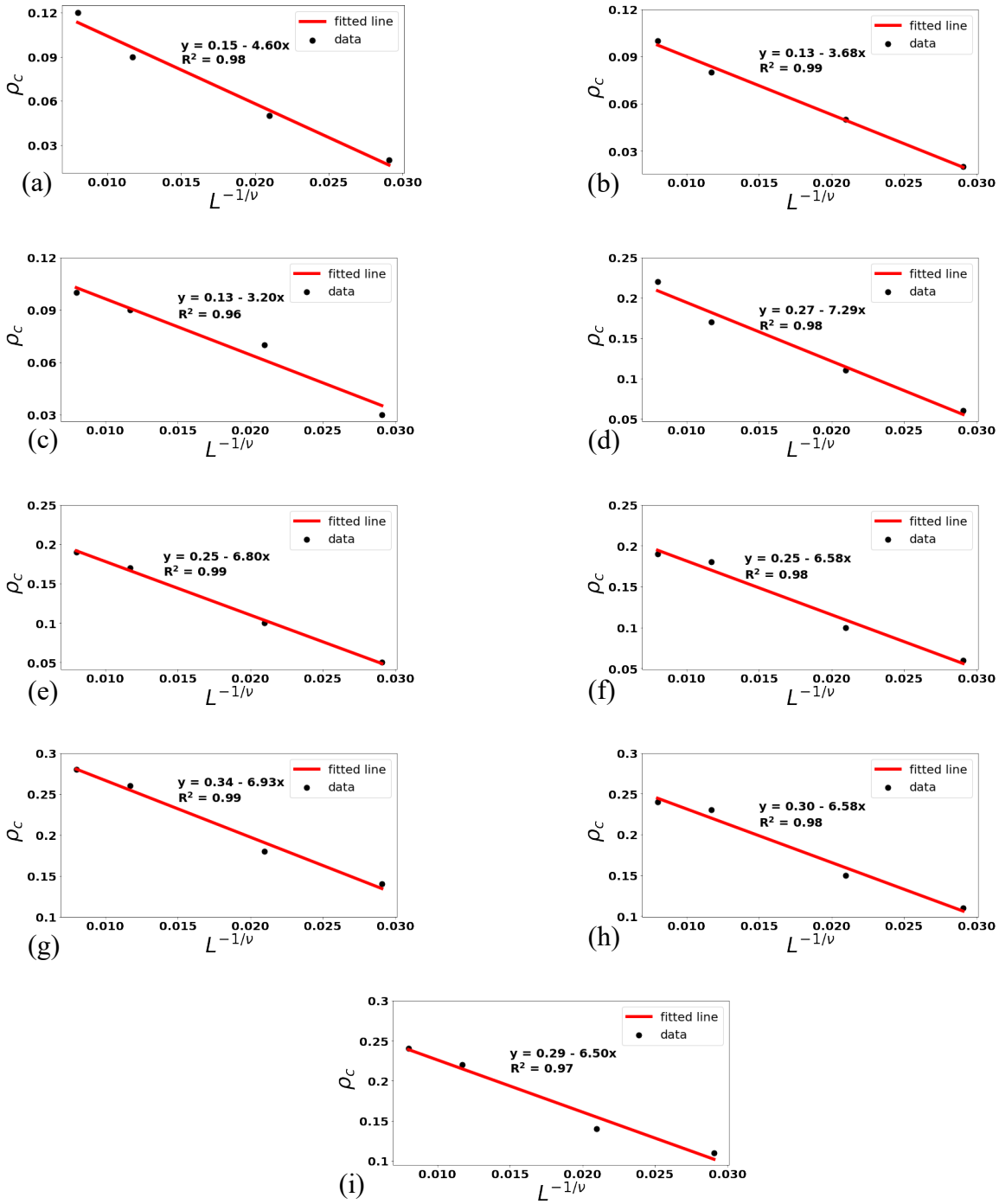

Figure 8. Relationship between critical fracture density (ρ_c) and system size (Fitted red line indicates the finite-size scaling equation (Eq. 7)). (a) plot generated at α = 2.5 and log (k_f/k_m) = 2, (b) plot generated at α = 2.5 and log (k_f/k_m) = 4, (c) plot generated at α = 2.5 and

log (k_f/k_m) = 6, (d) plot generated at α = 2.0 and log (k_f/k_m) = 2, (e) plot generated at α = 2.0 and log (k_f/k_m) = 4, (f) plot generated at α = 2.0 and log (k_f/k_m) = 6, (g) plot generated at α = 1.5 and log (k_f/k_m) = 2, (h) plot generated at α = 1.5 and log (k_f/k_m) = 4, and (i) plot generated at α = 1.5 and log (k_f/k_m) = 6.

The optimized value of $\rho_c(L \rightarrow \infty)$ for all nine matrix-fracture systems is presented in Table 4. Similar to our previous ρ_c results, we found that the value of $\rho_c(L \rightarrow \infty)$ increased with the increase of α. Results tabulated in Table 4 also show that the value of $\rho_c(L \rightarrow \infty)$ in systems with the same α (particularly α = 1.5 and 2) did not change from log(k_f/k_m) = 2 to 6, which means that $\rho_c(L \rightarrow \infty)$ is dependent on the value of α than log(k_f/k_m). This finding further strengthens the outcome of our regression analysis used to explain variability in ρ_c, which found no statistical evidence of a relationship between ρ_c and log(k_f/k_m).

Table 4. Summary of ρ_c and t for infinitely large matrix-fracture systems.

Reservoir	$\log(k_f/k_m)$	α	$\rho_c(L \rightarrow \infty)$	t_{inf}
1	2	1.5	0.15	2.20
2	4	1.5	0.13	1.91
3	6	1.5	0.13	1.84
4	2	2.0	0.25	2.06
5	4	2.0	0.24	1.72
6	6	2.0	0.24	1.60
7	2	2.5	0.34	1.70
8	4	2.5	0.30	1.61
9	6	2.5	0.29	1.60

- t for infinitely large systems

Matyka et al. (2008) proposed an empirical relationship to study the scale dependence of tortuosity and determine its value for an infinitely-large medium. Similarly, we applied the following exponential relationship to explain the scale dependency of the exponent t and extrapolate its value for $L \to \infty$

$$t(L) = t_{inf} + b\exp(-cL) \quad (11)$$

where b and c are two constant coefficients and t_{inf} is the value of t for an infinite-sized matrix-fracture system. In Eq. (11), as L approaches infinity, t tends to t_{inf}.

We plotted t against L and apply Eq. (11) to fit the data. Results presented in Fig. 9 show that Eq. (11) fit the $t - L$ data well with an average R^2 value of 0.99. We also listed the value of t_{inf} for all nine matrix-fracture systems in Table 4. As can be seen, the value of t_{inf} decreased as the value of $\log(k_f/k_m)$ increased. We also observed that greater α values corresponded to smaller t_{inf} values, which is in well accord with our regression-based results presented in Eq. (9).

The decreasing trend of t with increasing L shown in Fig. 9 is consistent with the results of Tremblay and Machta (1989) who theoretically demonstrated that the scaling exponent t is scale dependent and numerically showed that its value decreased as L increased in two and three dimensions.

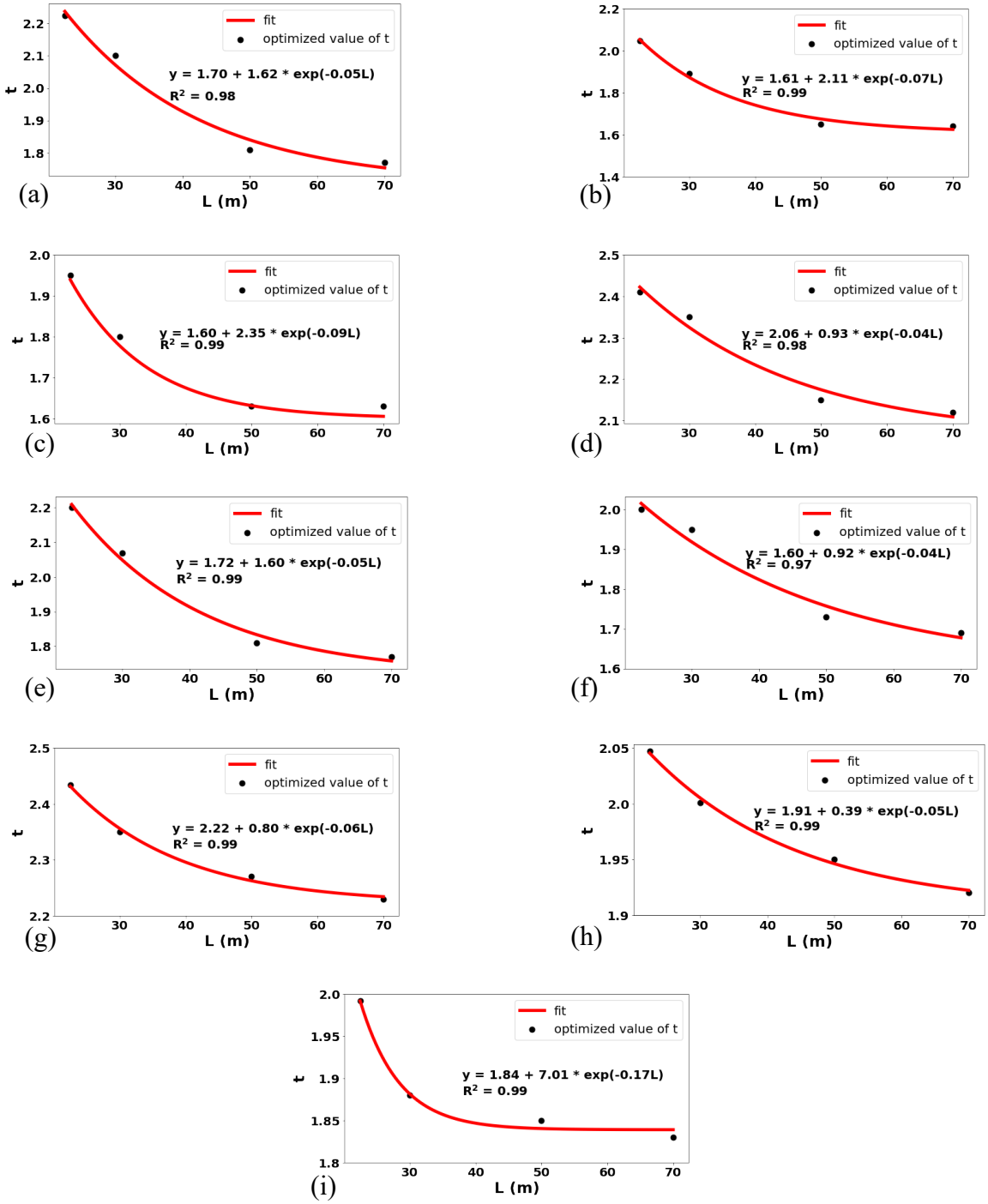

Figure 9. The scaling exponent t against the system size L for (a) $\alpha = 2.5$ and $\log(k_f/k_m) = 2$, (b) $\alpha = 2.5$ and $\log(k_f/k_m) = 4$, (c) $\alpha = 2.5$ and $\log(k_f/k_m) = 6$, (d) $\alpha = 2.0$ and $\log(k_f/k_m) = 2$, (e) $\alpha = 2.0$ and $\log(k_f/k_m) = 4$, (f) $\alpha = 2.0$ and $\log(k_f/k_m) = 6$, (g) $\alpha = 1.5$ and

log (k_f/k_m) = 2, (h) α = 1.5 and log (k_f/k_m) = 4, and (i) α = 1.5 and log (k_f/k_m) = 6. The red line represents the fit of Eq. (8) to the data.

- Estimating the $k_{eff} - \rho$ curve for infinitely-large systems

To extend our results to infinitely-large systems, we replaced ρ_c and t in Eq. (7) with the calculated values of $\rho_c(L \rightarrow \infty)$ and t_{inf} reported in Table 4 and determined the k_{eff} at various ρ values via the P-EMA. Results are given in Fig. 10 for α = 1.5, 2.0, and 2.5, and log(k_f/k_m) = 2, 4, and 6. Fig. 10 also shows the $k_{eff} - \rho$ curves corresponding to L = 22.5, 30, 50, and 70 m. As can be observed, $k_{eff} - \rho$ curves for L = 22.5m, 30m, 50m, 70m, and infinity and log(k_f/k_m) = 2 are almost inseparable (Figs. 10a, 10d, and 10g) meaning that the effect of scale on k_{eff} was negligible when log(k_f/k_m) = 2. However, due to higher level of heterogeneity, the influence of scale became more substantial as the value of log(k_f/k_m) increased from 2 to 6, which further reveals the impact of log (k_f/k_m) on scale dependence of k_{eff}. This means that the greater the log(k_f/k_m) value, the more pronounced the scale dependence of k_{eff} in matrix-fracture systems. We also found that the value of log(k_f/k_m) also impacted the shape of the $k_{eff} - \rho$ curve. At log(k_f/k_m) = 2, k_{eff} was observed to keep increasing at $\rho \geq \rho_c$ until k_{eff} reaches k_f near $\rho = 1$, resulting in an increasing concave upward trend. As the value of log(k_f/k_m) increased to 4 and 6, the magnitude of increase in k_{eff} at $\rho \geq \rho_c$ was found to start becoming smaller at some ρ value, typically around ρ > 0.5, until the curve flattens out as k_{eff} approaches k_f resulting in a sigmoidal shape.

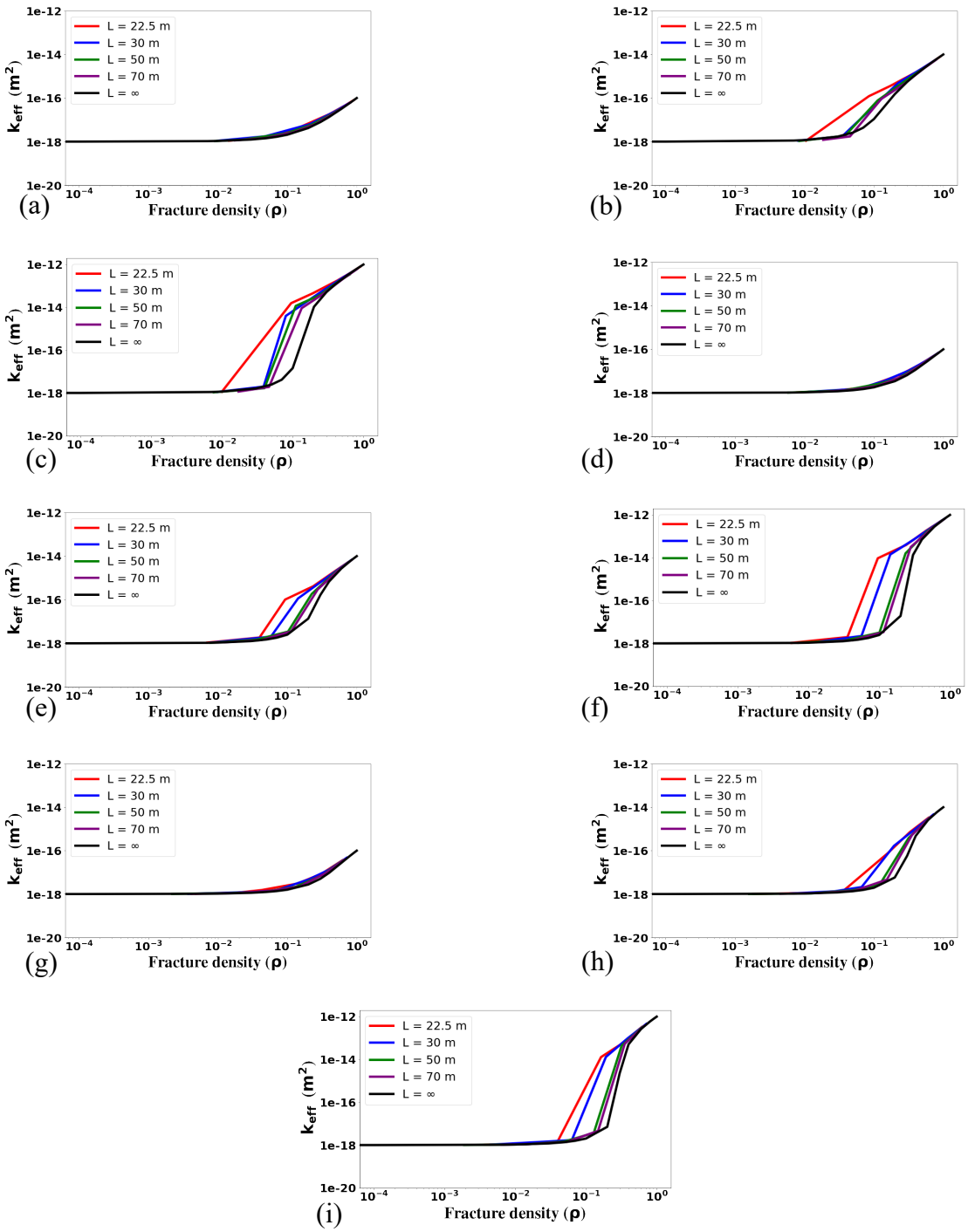

Figure 10. Scale dependence of the effective permeability, k_{eff}, in matrix-fracture systems with (a) α = 2.5 and log (k_f/k_m) = 2, (b) α = 2.5 and log (k_f/k_m) = 4, (c) α = 2.5 and log (k_f/k_m) = 6, (d) α = 2.0 and log (k_f/k_m) = 2, (e) α = 2.0 and log (k_f/k_m) = 4, (f) α = 2.0 and log (k_f/k_m) = 6, (g) α = 1.5 and log (k_f/k_m) = 2, (h) α = 1.5 and log (k_f/k_m) = 4, and (i) α =

1.5 and log (k_f/k_m) = 6. The $k_{eff} - \rho$ curves corresponding to infinitely-large systems ($L \rightarrow \infty$) were determined using the P-EMA and $\rho_c(L \rightarrow \infty)$ and t_{inf} values reported in Table 4.

Chapter 5 - Conclusions

In this study, we investigated the effective permeability, k_{eff}, in the matrix-fracture systems by means of numerical simulations and theoretical modeling. We assumed that the fracture length distribution followed the truncated power-law probability density function with exponent α = 1.5, 2, and 2.5 and generated nine matrix-fracture systems based on field observations. More specifically, we set minimum and maximum fracture lengths equal to 0.02 and 20 m, respectively, and considered $\log(k_f/k_m)$ = 2, 4, and 6 (where k_f and k_m are respectively fracture and matrix permeability values). To address the effect of scale, four system sizes L = 22.5, 30, 50, and 70 m were considered, and, overall, 36 matrix-fracture systems were studied. We numerically simulated fluid flow at six fracture densities by solving Richards' equation. The simulations were iterated at least 10 times at each fracture density with more than 5000 iterations in total. The simulated $k_{eff} - \rho$ curves were then averaged and fit by the percolation-based effective-medium approximation (P-EMA) and its parameters, ρ_c (critical fracture density) and t (scaling exponent), were optimized. Results demonstrated that both ρ_c and t were scale dependent. We also found that the effect of scale was more significant in systems with greater $\log(k_f/k_m)$ values (= 4 and 6). Our numerical simulations indicated that the P-EMA parameters ρ_c and t depended on the matrix-fracture characteristics, such as α, $\log(k_f/k_m)$, maximum fracture length (l_{max}), and system size (L). Using the stepwise multiple-linear regression analysis, we developed models that linked ρ_c and t to other matrix-fracture properties. We found good performance of the regression-based model to predict ρ_c and t for different realization of matrix-fracture systems. We also extended our simulations to infinitely-large fractured reservoirs by determining the values of ρ_c and t at $L \to \infty$.

References

Azizmohammadi, S., & Matthäi, S. K. (2017). Is the permeability of naturally fractured rocks scale dependent? Water Resources Research, 53(9), 8041-8063.

Baghbanan, A., & Jing, L. (2007). Hydraulic properties of fractured rock masses with correlated fracture length and aperture. International Journal of Rock Mechanics and Mining Sciences, 44(5), 704-719.

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.

Berkowitz, B., Bour, O., Davy, P., & Odling, N. (2000). Scaling of fracture connectivity in geological formations. Geophysical Research Letters, 27(14), 2061-2064.

Berrone, S., Canuto, C., Pieraccini, S., & Scialò, S. (2018). Uncertainty quantification in discrete fracture network models: Stochastic geometry. Water Resources Research, 54(2), 1338-1352.

Best, M. E., & Katsube, T. J. (1995). Shale permeability and its significance in hydrocarbon exploration. The Leading Edge, 14(3), 165-170.

Bogdanov, I. I., Mourzenko, V. V., Thovert, J. F., & Adler, P. M. (2003). Effective permeability of fractured porous media in steady state flow. Water Resources Research, 39(1) 1023.

Bonnet, E., Bour, O., Odling, N. E., Davy, P., Main, I., Cowie, P., & Berkowitz, B. (2001). Scaling of fracture systems in geological media. Reviews of Geophysics, 39(3), 347-383.

Cacas, M. C., Ledoux, E., de Marsily, G., Tillie, B., Barbreau, A., Durand, E., ... & Peaudecerf, P. (1990). Modeling fracture flow with a stochastic discrete fracture network: calibration and validation: 1. The flow model. Water Resources Research, 26(3), 479-489.

Deprez, N., McLachlan, D. S., & Sigalas, I. (1988). The measurement and comparative analysis of the electrical and thermal conductivities, permeability and Young's modulus of sintered nickel. Solid State Communications, 66(8), 869-872.

Dverstorp, B., Andersson, J., & Nordqvist, W. (1992). Discrete fracture network interpretation of field tracer migration in sparsely fractured rock. Water Resources Research, 28(9), 2327-2343.

de Dreuzy, J. R., Davy, P., & Bour, O. (2000). Percolation parameter and percolation-threshold estimates for three-dimensional random ellipses with widely scattered distributions of eccentricity and size. Physical Review E, 62(5), 5948-5952.

de Dreuzy, J. R., Davy, P., & Bour, O. (2001a). Hydraulic properties of two-dimensional random fracture networks following a power law length distribution: 1. Effective connectivity. Water Resources Research, 37(8), 2065-2078.

de Dreuzy, J. R., Davy, P., & Bour, O. (2001b). Hydraulic properties of two-dimensional random fracture networks following a power law length distribution: 2. Permeability of networks based on lognormal distribution of apertures. Water Resources Research, 37(8), 2079-2095.

de Dreuzy, J. R., Davy, P., & Bour, O. (2002). Hydraulic properties of two-dimensional random fracture networks following power law distributions of length and aperture. Water Resources Research, 38(12), 12-1.

de Dreuzy, J. R., Méheust, Y., & Pichot, G. (2012). Influence of fracture scale heterogeneity on the flow properties of three-dimensional discrete fracture networks (DFN). Journal of Geophysical Research: Solid Earth, 117(B11).

Deutsch, C. (1989). Calculating effective absolute permeability in sandstone/shale sequences. SPE Formation Evaluation, 4(03), 343-348.

Durlofsky, L. J. (1991). Numerical calculation of equivalent grid block permeability tensors for heterogeneous porous media. Water Resources Research, 27(5), 699-708.

Ebigbo, A., Lang, P. S., Paluszny, A., & Zimmerman, R. W. (2016). Inclusion-based effective medium models for the permeability of a 3D fractured rock mass. Transport in Porous Media, 113, 137-158.

Forstner, S. R., & Laubach, S. E. (2022). Scale-dependent fracture networks. Journal of Structural Geology, 165, 104748.

Ghanbarian, B., & Daigle, H. (2016). Thermal conductivity in porous media: Percolation-based effective-medium approximation. Water Resources Research, 52(1), 295-314.

Hunt, A., Ewing, R., & Ghanbarian, B. (2014). Percolation Theory for Flow in Porous Media (Vol. 880). Springer.

Hyman, J. D., Karra, S., Makedonska, N., Gable, C. W., Painter, S. L., & Viswanathan, H. S. (2015). dfnWorks: A discrete fracture network framework for modeling subsurface flow and transport. Computers & Geosciences, 84, 10-19.

Hyman, J. D., Karra, S., Carey, J. W., Gable, C. W., Viswanathan, H., Rougier, E., & Lei, Z. (2018). Discontinuities in effective permeability due to fracture percolation. Mechanics of Materials, 119, 25-33.

Ji, S. H., Lee, K. K., & Park, Y. C. (2004). Effects of the correlation length on the hydraulic parameters of a fracture network. Transport in Porous Media, 55(2), 153-168.

Jourde, H., Aydin, A., & Durlofski, L. (2002). Upscaling permeability of fault zones in porous sandstone: from field measurement to numerical modelling. AAPG Bull, 86(7), 1187-1200.

Journel, A. G., Deutsch, C., & Desbarats, A. J. (1986). Power averaging for block effective permeability. In *SPE California Regional Meeting*. OnePetro.

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

Lei, Q., Latham, J. P., Tsang, C. F., Xiang, J., & Lang, P. (2015). A new approach to upscaling fracture network models while preserving geostatistical and geomechanical characteristics. Journal of Geophysical Research: Solid Earth, 120(7), 4784-4807.

Lichtner, P. C., Hammond, G. E., Lu, C., Karra, S., Bisht, G., Andre, B., ... & Kumar, J. (2015). *PFLOTRAN user manual: A massively parallel reactive flow and transport model for*

describing surface and subsurface processes (No. LA-UR-15-20403). Los Alamos National Lab. (LANL), Los Alamos, NM (United States); Sandia National Lab. (SNL-NM), Albuquerque, NM (United States); Lawrence Berkeley National Lab. (LBNL), Berkeley, CA (United States); Oak Ridge National Lab. (ORNL), Oak Ridge, TN (United States); OFM Research, Redmond, WA (United States).

Long, J. C., & Witherspoon, P. A. (1985). The relationship of the degree of interconnection to permeability in fracture networks. Journal of Geophysical Research: Solid Earth, 90(B4), 3087-3098.

Lough, M. F., Lee, S. H., & Kamath, J. (1997). A new method to calculate effective permeability of gridblocks used in the simulation of naturally fractured reservoirs. SPE Reservoir Engineering, 12(03), 219-224.

Madadi, M., VanSiclen, C. D., & Sahimi, M. (2003). Fluid flow and conduction in two-dimensional fractures with rough, self-affine surfaces: A comparative study. Journal of Geophysical Research: Solid Earth, 108(B8), 2396.

Maillot, J., Davy, P., Le Goc, R., Darcel, C., & De Dreuzy, J. R. (2016). Connectivity, permeability, and channeling in randomly distributed and kinematically defined discrete fracture network models. Water Resources Research, 52(11), 8526-8545.

Matyka, M., Khalili, A., & Koza, Z. (2008). Tortuosity-porosity relation in porous media flow. Physical Review E, 78(2), 026306.

McLachlan, D. S. (1987). An equation for the conductivity of binary mixtures with anisotropic grain structures. J. Phys. C Solid State Phys., 20, 865–877.

McLachlan, D. S. (1988). Measurement and analysis of a model dual-conductivity medium using a generalised effective-medium theory. J. Phys. C Solid State Phys., 21, 1521-1532.

McLachlan, D. S. (2021). The percolation exponents for electrical and thermal conductivities and the permittivity and permeability of binary composites. Physica B: Condensed Matter, 606, 412658.

Mourzenko, V. V., Thovert, J. F., & Adler, P. M. (2005). Percolation of three-dimensional fracture networks with power-law size distribution. Physical Review E, *72*(3), 036103.

Mourzenko, V. V., Thovert, J. F., & Adler, P. M. (2011). Permeability of isotropic and anisotropic fracture networks, from the percolation threshold to very large densities. Physical Review E, 84(3), 036307.

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.

Nakashima, T., Sato, K., Arihara, N., & Yazawa, N. (2000). Effective permeability estimation for simulation of naturally fractured reservoirs. In *SPE Asia Pacific Oil and Gas Conference and Exhibition*. OnePetro.

Neuman, S. P. (1994). Generalized scaling of permeabilities: Validation and effect of support scale. Geophysical Research Letters, 21(5), 349-352.

Nordqvist, A. W., Tsang, Y. W., Tsang, C. F., Dverstorp, B., & Andersson, J. (1992). A variable aperture fracture network model for flow and transport in fractured rocks. Water Resources Research, 28(6), 1703-1713.

Oda, M. A. S. A. N. O. B. U. (1985). Permeability tensor for discontinuous rock masses. Geotechnique, 35(4), 483-495.

Painter, S., & Cvetkovic, V. (2005). Upscaling discrete fracture network simulations: An alternative to continuum transport models. Water Resources Research, 41(2) W02002.

Sævik, P. N., Berre, I., Jakobsen, M., & Lien, M. (2013). A 3D computational study of effective medium methods applied to fractured media. Transport in Porous Media, 100, 115-142.

Sahimi, M., & Mukhopadhyay, S. (1996). Scaling properties of a percolation model with long-range correlations. Physical Review E, 54(4), 3870.

Sahimi, M. (2011). Flow and transport in porous media and fractured rock: from classical methods to modern approaches. John Wiley & Sons.

Stauffer, D., & Aharony, A. (2018). *Introduction to percolation theory*. Taylor & Francis.

Sweeney, M. R., Gable, C. W., Karra, S., Stauffer, P. H., Pawar, R. J., & Hyman, J. D. (2020). Upscaled discrete fracture matrix model (UDFM): an octree-refined continuum representation of fractured porous media. Computational Geosciences, 24(1), 293-310.

Teufel, L. W., Rhett, D. W., Farrell, H. E., & Lorenz, J. C. (1993). Control of fractured reservoir permeability by spatial and temporal variations in stress magnitude and orientation. In *SPE Annual Technical Conference and Exhibition*. OnePetro.

Tremblay, A. M., & Machta, J. (1989). Finite-size effects in continuum percolation. Physical Review B, 40(7), 5131.

Vermilye, J. M., & Scholz, C. H. (1995). Relation between vein length and aperture. Journal of Structural Geology, 17(3), 423-434.

Viswanathan, H. S., Hyman, J. D., Karra, S., O'Malley, D., Srinivasan, S., Hagberg, A., & Srinivasan, G. (2018). Advancing graph-based algorithms for predicting flow and transport in fractured rock. Water Resources Research, 54(9), 6085-6099.

Viswanathan, H. S., Ajo-Franklin, J., Birkholzer, J. T., Carey, J. W., Guglielmi, Y., Hyman, J. D., ... & Tartakovsky, D. M. (2022). From fluid flow to coupled processes in fractured rock: Recent advances and new frontiers. Reviews of Geophysics, 60(1), e2021RG000744.

Wang, F. P., Reed, R. M., John, A., & Katherine, G. (2009). Pore networks and fluid flow in gas shales. In *SPE annual technical conference and exhibition*. OnePetro.

Zanon, S., Nguyen, H., & Deutsch, C. V. (2002). Power Law Averaging Revisited. *Centre for Computational Geostatistics Report Four, University of Alberta, Edmonton, Canada*.

Zhu, J. (2019). Impact of scaling break and fracture orientation on effective permeability of fractal fracture network. *SN Applied Sciences*, *1*(1), 4.

Zhu, W., Khirevich, S., & Patzek, T. W. (2021). Impact of fracture geometry and topology on the connectivity and flow properties of stochastic fracture networks. Water Resources Research, 57(7), e2020WR02

Appendix A - Discrete Fracture Network (DFN) generation script

```
/=================================================================
==========
//Gereral Options & Fracture Network Parameters:

stopCondition: 0
   /* 0: stop once nPoly fractures are accepted (Defined below)
      1: stop once all family's p32 values are equal or greater than the families
               target p32 values (defined in stochastic family sections)
   */

nPoly: 15000      /* nPoly means number of fractures defined */
    /* Used when stopCondition = 0
         Total number of fractures you would like to have
         in the domain you defined. The program will complete
         once you have nPoly number of fractures,
         maxPoly number of polygon/fracture rejections,
         rejPoly number of rejections in a row, or reach a
         specified fracture cluster size if using
         stoppingParameter = -largestSize  */


outputAllRadii: 0
/* 0: Do not output all radii file.
   1: Include file of all raddii (acepted+rejected fractures)
      in output files.
*/


domainSize: {70,70,70}
```

```
    /* Mandatory Parameter.
       Creates a domain with dimension x*y*z centered at the origin.*/

numOfLayers: 0    //number of layers


layers:
{-500,0}
{0,500}

/*  Layers need to be listed line by line
   Format: {minZ, maxZ}

   The first layer listed is layer 1, the second is layer 2, etc
   Stochastic families can be assigned to theses layers (see stochastic
   shape familiy section)
*/


numOfRegions: 0 // Number of regions
regions:
{}

/*  Regions need to be listed line by line
   Format: {minX, maxX, minY, maxY, minZ, maxZ}

   The first region listed is region 1, the second is region 2, etc
   Stochastic families can be assigned to theses layers (see stochastic
   shape family section)
*/
```

```
orientationOption: 0
/* Fracture Orientation Option
   0 : Spherical Coordinates
   1 : Trend / Plunge
   2 : Dip / Strike
*/




h: 0.050
  /* Minimum fracture length scale(meters)
    Any fracture with a feature, such as and intersection, of less than h will be rejected. */



//=============================================================================
======//
/* Fracture Network Parameters:                                    */

tripleIntersections: 1
/* Options:     0: Off
                1: On    */

printRejectReasons: 0
/* Useful in debugging,
   This option will print all fracture rejection reasons as they occur.
       0: disable
       1: print all rejection reasons to screen
*/
```

```
visualizationMode: 1
/* Options: 0 or 1
   Used during meshing:
   0: creates a fine mesh, according to h parameter;
   1: produce only first round of triangulations. In this case no
   modeling of flow and transport is possible. */

seed: 92731535
    //Seed for random generator.



domainSizeIncrease: {0,0,0}
        //temporary size increase for inserting fracture centers outside domain
        //increases the entire width by this ammount. So, {1,1,1} will increase
        //the domain by adding .5 to the +x, and subbtracting .5 to the -x, etc



keepOnlyLargestCluster: 0
    /* 0 = Keep any clusters which connects the specified
          boundary faces in boundaryFaces option below

       1 = Keep only the largest cluster which connects
          the specified boundary faces in boundaryFaces option below
    */

ignoreBoundaryFaces: 1
/*
     0 = use boundaryFaces option below

     1 = ignore boundaryFaces option and keep all clusters and
     will still remove fractures with no intersections
```

```
*/


boundaryFaces: {1,1,0,0,0,0}
/*  DFN will only keep clusters with connections to domain boundaries which are set to 1:

   boundaryFaces[0] = +X domain boundary
   boundaryFaces[1] = -X domain boundary
   boundaryFaces[2] = +Y domain boundary
   boundaryFaces[3] = -Y domain boundary
   boundaryFaces[4] = +Z domain boundary
   boundaryFaces[5] = -Z domain boundary

   Be sure to set ignoreBoundaryFaces to 0 when using this feature.
*/


rejectsPerFracture: 10  /*If fracture is rejected, it will be re-translated to a new
                        position this number of times.

                        This helps hit distribution targets for stochastic families
                        (Set to 1 to ignore this feature)    */




//=============================================================================
=======
//                Shape and Probability Parameters
```

```
//=============================================================================
=======

//user rectangles and user Ellipses defined in their cooresponding files

famProb: {.5,.5}
/* Probability of occurrence of each family of randomly distrubuted rectangles
  and ellipses.
  User-ellipses and user-rectangles insertion will be attempted with 100%
  likelihood, but with possability they may be rejected.
  The famProb elements should add up to 1.0 (for %100).
  The probabilities are listed in order of families starting with all stochastic
  ellipses, and then all stochastic rectangles.

  For example:
      If  then there are two ellipse families, each with probabiliy .3,
      and two rectangle families, each with probabiliy .2, famProb will be:
      famProb: {.3,.3,.2,.2} Notice: famProb elements add to 1
*/




/*=============================================================================
========*/
//=============================================================================
=======
//                  Elliptical Fracture Options
//     NOTE: Number of elements must match number of ellipse families
//          (first number in nShape input parameter)
```

```
//=========================================================================
=======
/*=========================================================================
========*/

//Number of ellipse families
nFamEll: 0
   //Having this option = 0 will ignore all rectangle family variables

eLayer: {0,0}
   /* Defines which domain the family belongs to.
    Layer 0 is the entire domain.
    Layers numbered > 0 corresponds to layers defined above
    1 corresponds to the first layer listed, 2 is the next layer listed, etc */

eRegion: {0,0}
   /* Defines which domain the family belongs to.
    Region 0 is the entire domain.
    Regions numbered > 0 corresponds to layers defined above
    1 corresponds to the first region listed, 2 is the next region listed, etc */


//edist is a mandatory parameter if using statistically generated ellipses
edistr: {2,3}   /* Ellipse statistical distribution options:
                   1 - lognormal distribution
                   2 - truncated power law distribution
                   3 - exponential distribution
                   4 - constant */


ebetaDistribution: {1,1}   /* Beta is the rotation around the polygon's normal
```

vector, with the polygon centered on x-y plane at the orgin

0 - uniform distribution [0, 2PI]
1 - constant angle (specefied below by "ebeta") */

e_p32Targets: {.1,.1}
/* Elliptical families target fracture intensity per family.
When using stopCondition = 1 (defined at the top of the input file), families will
be inserted untill the families desired fracture intensity has been reached.
Once all families desired fracture intensity has been met, fracture generation will
be complete.
*/

//===
=======
// Parameters used by all stochastic ellipse families
// Mandatory Parameters if using statistically generated ellipses

easpect: {1,1} /* Aspect ratio. Used for lognormal and truncated
power law distribution. */

enumPoints: {12, 12} /*Number of vertices used in creating each elliptical
fracture family. Number of elements must match number
of ellipse families (first number in nShape) */

eAngleOption: 0 /* All angles for ellipses:
0 - degrees
0 - Radians (Must use numerical value for PI) */

etheta: {-45, 45} /*Ellipse fracture orientation.

```
                The angle the normal vector makes with the z-axis */

ephi: {0,0}   /* Ellipse fracture orientation.
                The angle the projection of the normal onto the x-y plane
                makes with the x-axis */

ebeta: {0, 0}   /* rotation around the normal vector */


ekappa: {8,8}  /*Parameter for the fisher distribnShaprutions. The bigger, the more
                  similar (less diverging) are the elliptical familiy's
                  normal vectors */

//=============================================================================
=======
// Options Specific For Ellipse Lognormal Distribution (edistr=1):
// Mandatory Parameters if using ellispes with lognormal distribution

//          NOTE: Number of elements must match number of
//                ellipse families (first number in nShape)

eLogMean: {2}  //Mean value For Lognormal Distribution.

eLogMax: {100}
eLogMin: {1}

esd: {.5} // Standard deviation for lognormal distributions of ellipses

//=============================================================================
=======
//     Options Specific For Ellipse Exponential Distribution (edistr=3):
```

```
//     Mandatory Parameters if using ellispes with exponential distribution


eExpMean: {2}  //Mean value for Exponential  Distribution
eExpMax: {3}  //Mean value for Exponential  Distribution
eExpMin: {1}  //Mean value for Exponential  Distribution

//=========================================================================
=======
//    Options Specific For Constant Size of ellipses (edistr=4):

econst: {10, 10, 10}  // Constant radius, defined per family

//=========================================================================
=======
// Options Specific For Ellipse Truncated Power-Law Distribution (edistr=2)
// Mandatory Parameters if using ellipses with truncated power-law dist.

// NOTE: Number of elements must match number
//       of ellipse families (first number in nShape)

emin: {1} // Minimum radius for each ellipse family.
          // For power law distributions.

emax: {6}  // Maximum radius for each ellipse family.
               // For power law distributions.

ealpha: {2.4} // Alpha. Used in truncated power-law
                    // distribution calculation
```

```
/*==========================================================================
==============*/
/*==========================================================================
================*/
/*                        Rctangular Fractures Options                      */
/* NOTE: Number of elements must match number of rectangle families         */
/*       (second number in nShape parameter)                                */
/*==========================================================================
==============*/
/*==========================================================================
================*/

//Number of rectangle families
nFamRect: 0
   //Having this option = 0 will ignore all rectangle family variables



rLayer: {0,0}
   /* Defines which domain the family belongs to.
    Layer 0 is the entire domain.
    Layers numbered > 0 corresponds to layers defined above
    1 corresponds to the first layer listed, 2 is the next layer listed, etc */

rRegion: {}
   /* Defines which domain the family belongs to.
    Region 0 is the entire domain.
    Regions numbered > 0 correspond to layers defined above
    1 corresponds to the first region listed, 2 is the next region listed, etc */
```

```
/*rdist is a mandatory parameter if using statistically generated rectangles */
rdistr: {2,3}   /*  Rectangle statistical distribution options:
                    1 - lognormal distribution
                    2 - truncated power law distribution
                    3 - exponential distribution
                    4 - constant */

rbetaDistribution: {1,1}   /* Beta is the rotation/twist about the z axis
                with the polygon centered on x-y plane at the origin
                before rotation into 3d space

                0 - uniform distribution [0, 2PI]
                1 - constant angle (specified below by "rbeta")

             */

r_p32Targets: {.1,.1}
/* Rectangle families target fracture intensity per family.
When using stopCondition = 1 (defined at the top of the input file), families will
be inserted until the families desired fracture intensity has been reached.
Once all families desired fracture intensity has been met, fracture generation will
be complete.
*/

//=============================================================================
========
// Parameters used by all stochastic rectangle families
// Mandatory Parameters if using statistically generated rectangles
```

```
raspect: {1,1}  /* Aspect ratio */

rAngleOption: 0     /* All angles for rectangles:
                    0 - degrees
                    1 - radians (must be numerical value, cannot use "Pi") */

rtheta: {-45,45} /*Rectangle fracture orientation.
                     The angle the normal vector makes with the z-axis */

rphi: {0,45} /* Rectangle fracture orientation.
              The angle the projection of the normal onto the x-y
              plane makes with the x-axis */

rbeta: {0,0}   /* rotation around the normal vector */

rkappa: {8,8}  /*Parameter for the fisher distributions. The bigger,
                        the more similar (less diverging) are the rectangle
                        familiy's normal vectors  */

//=================================================================
=========
// Options Specific For Rectangle Lognormal Distribution (rdistr=1):
// Mandatory Parameters if using rectangles with lognormal distribution

rLogMean: {1.6}   /*For Lognormal Distribution.
                  Mean radius (1/2 rectangle length) in
                  lognormal distribution for rectangles. */


rLogMax: {100}
```

```
rLogMin: {1}

rsd: {.4}     /* Standard deviation for lognormal distributions of
                 rectangles */

//=================================================================
=========
// Options Specific For Rectangle Truncated Power-Law Distribution (rdistr=2):
// Mandatory Parameters if using rectangles with power-law distribution

 rmin: {1,1}         /* Minimum radius for each rectangle family.
                        For power law distributions. */

 rmax: {6,5}    /* Maximum radius for each rectangle family.
                   For power law distributions. */

 ralpha: {2.5}    // Alpha. Used in truncated power-law
                 // distribution calculation



/*=================================================================
========*/
/* Options Specific For Rectangle Exponential Distribution (edistr=3):       */
/* Mandatory Parameters if using rectangules with exponential distribution   */

rExpMean: {2}  //Mean value for Exponential  Distribution
rExpMax: {100}
rExpMin: {1}

/*=================================================================
========*/
```

```
/* Options Specific For Constant Size of rectangles (edistr=4):              */

rconst: {4,4}  // Constant radius, defined per rectangular family

/*============================================================================
========*/
/*============================================================================
========*/
/* User-Specified Ellipses                                                   */
/* Mandatory Parameters if using user-ellipses                               */
/* NOTE: Number of elements must match number of user-ellipse families       */
/*(third number in nShape parameter)                                         */
/*============================================================================
========*/
/* NOTE: Only one user-ellipse is placed into the domain per defined
        user-ellipse, with possibility of being rejected  */


userEllipsesOnOff: 0    //0 - User Ellipses off
                        //1 - User Ellipses on

UserEll_Input_File_Path: ./TestCases/test/uEllInput.dat

/*============================================================================
========*/
/*============================================================================
========*/
/*  User-Specified Ellipses                                                  */
/*  Mandatory Parameters if using user-ellipses                              */
/*  NOTE: Number of elements must match number of user-ellipse families.     */
/*  NOTE: Only one user-ellipse is placed into the domain per defined        */
```

```
/*        user-ellipse, with possibility of being rejected                  */
/*========================================================================
========*/
/*========================================================================
========*/

userEllByCoord: 0
/*  0 - User ellipses defined by coordinates off
    1 - User ellipses defined by coordinates on
*/

EllByCoord_Input_File_Path:
/home/jharrod/GitProjects/DFNGen/DFNC++Version/inputFiles/userPolygons/ellCoords.dat



/*========================================================================
========*/
/* User-Specified Rectangles                                              */
/* Mandatory Parameters if using user-rectangles                          */
/* NOTE: Number of elements must match number of user-ellipse families    */
/* (fourth number in nShape parameter)                                    */
/*========================================================================
========*/
/* NOTE: Only one user-rectangle is placed into the domain per defined
        user-rectangle, with possibility of being rejected  */


userRectanglesOnOff: 1    //0 - User Rectangles off
                          //1 - User Rectangles on
```

UserRect_Input_File_Path: /Users/jhyman/src/dfnworks-main/examples/octree/define_4_user_rects.dat

/*==========*/

/* If you would like to input user specified rectangles according to their
coordinates, you can use the parameter userDefCoordRec. In that case, all
of the user specified rectangles will have to be according to coordinates.
*/

userRecByCoord: 0

// 0 - user defined rectangles not used

// 1 - user defined rectangles used and defined by input file:

RectByCoord_Input_File_Path: ./inputFiles/userPolygons/rectCoords.dat

/*WARNING: userDefCoordRec can cause LaGriT errors because the polygon vertices are not put in clockwise or counter-clockwise order.
If errors (Usualy seg fualt during meshing if using LaGriT),
try to reorder the points till u get it right.
Also, coordinates must be co-planar */

/*==========*/

// Aperture [m]

/* Mandatory parameter, and can be specified in several ways:

- 1)meanAperture and stdAperture for using LogNormal distribution.
- 2)apertureFromTransmissivity, first transmissivity is defined, and then, using a cubic law, the aperture is calculated;
- 3)constantAperture, all fractures, regardless of their size, will have

```
  the same aperture value;
- 4)lengthCorrelatedAperture, aperture is defined as a function of fracture size*/

//NOTE: Only one aperture type may be used at a time

aperture: 3  //choise of aperture option described above

//(**** 1)meanAperture and stdAperture for using LogNormal distribution.********)
meanAperture:  -3 /*Mean value for aperture using
                          normal distribution */
stdAperture: 0.8  //Standard deviation

/*(****** 2)apertureFromTransmissivity, first transmissivity is defined,
  and then, using a cubic law, the aperture is calculated;***************/
apertureFromTransmissivity: {1.6e-9, 0.8}
    /* Transmissivity is calculated as transmissivity = F*R^k,
       where F is a first element in aperturefromTransmissivity,
       k is a second element and R is a mean radius of a polygon.
       Aperture is calculated according to cubic law as
       b=(transmissivity*12)^1/3 */

/*(****** 3)constantAperture, all fractures, regardless of their size,
   will have the same aperture value;    **********************************/

constantAperture: 0.00125  //Sets constant aperture for all fractures

/*(******** 4)lengthCorrelatedAperture, aperture is defined as a function of
       fracture size ******************/

lengthCorrelatedAperture: {5e-5, 0.5}
    /*Length Correlated Aperture Option:
```

```
    Aperture is calculated by: b=F*R^k,
    where F is a first element in lengthCorrelatedAperture,
    k is a second element and R is a mean radius of a polygon.*/


//=========================================================================
========
//Permeability
/* Options:
   0: Permeability of each fracture is a function of fracture aperture,
    given by k=(b^2)/12, where b is an aperture and k is permeability
   1: Constant permeabilty for all fractures */

permOption: 1  //See above for options

constantPermeability: 1e-12  //Constant permeability for all fractures

//=========================================================================
=========


// TODO: confirm with JDH
outputAcceptedRadiiPerFamily:1 /* output radii files for each
                        family containing the final radii chosen */
disableFram:0 /* 0 if FRAM (feature rejection algorithm for meshing)
            is disabled, 1 otherwise */
outputFinalRadiiPerFamily:1 /* output radii files
                          for each family containing the final radii chosen */
insertUserRectanglesFirst:1 /* 1 if user defined
                        rectangles should be inserted first, 0 otherwise */
forceLargeFractures:0 /* Force large fractures (fractures that X)  to be included in the network */
```

radiiListIncrease: 0.1 /* Increase the length of the radii list by this percentage */
removeFracturesLessThan: 0 /*Used to change the lower cutoff of fracture size*/

keepIsolatedFractures: 0
/* 0 - Remove any isolated fracture (not clusters)
1 - Keep all fractures in the domain
*/

/*==========
==========*/
/*==========
==========*/
/* */
/* User Polygon Defined By Coordinates */
/* */
/*==========
==========*/
/*==========
==========*/

userPolygonByCoord: 0
/* 0 - User defined polygon by coordinates off
1 - User defined polygon by coordinates on
*/

PolygonByCoord_Input_File_Path: ./

/*WARNING: userDefCoordRec can cause LaGriT errors because the polygon

vertices are not put in clockwise or counter-clockwise order.
If errors (Usualy seg fualt during meshing if using LaGriT),
try to reorder the points till u get it right.
Also, coordinates must be co-planar */

Appendix B - Fluid flow simulation script

```
# Sept 12, 2018
# Matthew Sweeney, Satish Karra, Jeffrey Hyman (LANL)
#================================================

SIMULATION
  SIMULATION_TYPE SUBSURFACE
  PROCESS_MODELS
    SUBSURFACE_FLOW flow
      MODE RICHARDS
    /
  /
END
SUBSURFACE



NUMERICAL_METHODS FLOW
  LINEAR_SOLVER
    SOLVER DIRECT
  /
END



#=========================== discretization
====================================

GRID
  TYPE unstructured_explicit full_mesh.uge
  GRAVITY 0.d0 0.d0 0.d0
END
```

```
#========================== fluid properties
================================
FLUID_PROPERTY
  DIFFUSION_COEFFICIENT 1.d-12
END

DATASET Permeability
  FILENAME mesh_permeability.h5
END

DATASET Porosity
  FILENAME mesh_porosity.h5
END

#========================== material properties
==============================
MATERIAL_PROPERTY matrix
  ID 1
  POROSITY DATASET Porosity
  TORTUOSITY 0.5d0
  CHARACTERISTIC_CURVES default
  PERMEABILITY
    DATASET Permeability
  /
END

MATERIAL_PROPERTY fracture
  ID 2
  POROSITY DATASET Porosity
```

```
  TORTUOSITY 0.5d0
  CHARACTERISTIC_CURVES default
  PERMEABILITY
    DATASET Permeability
  /
END

#========================== characteristic curves
============================
CHARACTERISTIC_CURVES default
  SATURATION_FUNCTION VAN_GENUCHTEN
    M 0.5d0
    ALPHA  1.d-4
    LIQUID_RESIDUAL_SATURATION 0.1d0
    MAX_CAPILLARY_PRESSURE 1.d8
  /
  PERMEABILITY_FUNCTION MUALEM_VG_LIQ
    M 0.5d0
    LIQUID_RESIDUAL_SATURATION 0.1d0
  /
END

#========================== output options
===================================
OUTPUT
#  PERIODIC TIME 0.00002d0 second
#  FORMAT TECPLOT BLOCK
  PRINT_PRIMAL_GRID
  FORMAT VTK
  MASS_FLOWRATE
  MASS_BALANCE
```

```
  VARIABLES
    LIQUID_PRESSURE
    PERMEABILITY_X
    PERMEABILITY_Y
    PERMEABILITY_Z
    POROSITY
  /
END

#============================ times
=============================================

TIME
  INITIAL_TIMESTEP_SIZE  1.d-8 s
  FINAL_TIME 100000.0d0 s
  MAXIMUM_TIMESTEP_SIZE 100000.0d0 s
  STEADY_STATE
END

#============================ regions
=========================================

REGION All
  COORDINATES
    -5.d20 -5.d20 -5.d20
    5.d20 5.d20 5.d20
  /
END

REGION inflow
  FILE pboundary_left_w.ex
END
```

```
REGION outflow
  FILE pboundary_right_e.ex
END


#========================== flow conditions
===================================

FLOW_CONDITION initial
  TYPE
    PRESSURE dirichlet
  /
  PRESSURE 1.01325d6
END


FLOW_CONDITION outflow
  TYPE
    PRESSURE dirichlet
  /
  PRESSURE 1.d6
END

FLOW_CONDITION inflow
  TYPE
   PRESSURE dirichlet
  /
  PRESSURE 1.1d6
END

#========================== condition couplers
================================
```

```
# initial condition
INITIAL_CONDITION
  FLOW_CONDITION initial
  REGION All
END

BOUNDARY_CONDITION INFLOW
  FLOW_CONDITION inflow
  REGION inflow
END

BOUNDARY_CONDITION OUTFLOW
  FLOW_CONDITION outflow
  REGION outflow
END

#========================== stratigraphy couplers
==========================
STRATA
  FILE materials.h5
END

END_SUBSURFACE
```

Appendix C - Script for upscaling effective permeability k_{eff}

```
"""
.. file:: run_fehm.py
   :synopsis: run file for dfnWorks
   :version: 1.0
   :maintainer: Jeffrey Hyman, Carl Gable
.. moduleauthor:: Jeffrey Hyman <jhyman@lanl.gov>

"""
import os
from pydfnworks import *
import networkx as nx
import numpy as np
from h5py import *

def set_fracture_perm(mat_perm, frac_perm):

   fracs = np.genfromtxt("tag_frac.dat").astype(int)
   n = len(fracs)

   perm = mat_perm*np.ones(n)
   idx = np.where(fracs > 0)
   perm[idx] = frac_perm

   filename = 'materials.h5'
   h5file = File(filename,mode='w')

   # create integer array for cell ids
   iarray = np.arange(n,dtype='i4')
```

```
  # convert to 1-based
  iarray[:] += 1
  dataset_name = 'Cell Ids'
  h5dset = h5file.create_dataset(dataset_name, data=iarray)

  dataset_name = 'Permeability'
  h5dset = h5file.create_dataset(dataset_name, data=perm)

  h5file.close()



def check_percolation():
  G = DFN.create_graph("fracture","left","right")
  with open("percolation.dat","w") as fp:
    if (nx.has_path(G,'s','t')):
      fp.write("1")
      print("network connects boundaries")
    else:
      fp.write("0")
      print("network does not connects boundaries")



# Parameters to varry
mat_perm = 1e-16
mat_por = 0.1
frac_perm = 1e-12
l = 0.1
orl = 3

# these need to match your pflotran file
inflow_pressure = 2e6
```

```
outflow_pressure = 1e6
boundary_file = "pboundary_left_w.ex"
direction = "x"

## Create DFN
DFN = create_dfn()
DFN.make_working_directory()
DFN.check_input()
DFN.create_network()
check_percolation()
DFN.mesh_network(visual_mode=True)

# Mesh netork
DFN.set_flow_solver("PFLOTRAN")
DFN.inp_file = "octree_dfn.inp"

DFN.map_to_continuum(l,orl)
DFN.upscale(mat_perm,mat_por)

# Run flow. Uses a direct solve.
DFN.ncpu = 1
DFN.zone2ex(uge_file='full_mesh.uge',zone_file='all')
DFN.pflotran()
DFN.parse_pflotran_vtk_python()
DFN.pflotran_cleanup()
DFN.effective_perm(inflow_pressure, outflow_pressure, boundary_file, direction)
```

Appendix D - Script for automating simulation

```
import numpy as np
param1 = ['10','30','50','70','120','200']  # === nPoly to vary
param2 = ['1e-13','1e-11']    # === permeability ratio to vary
# param3 = list(np.arange(18)+1)

# funtion to read the simulation input files
def open_note(filename):
    with open(filename) as f:
        lines_0 = f.readlines()
    f.close()
    return(lines_0)

# Define simulation input files to automate
f1 = 'gen_4_user_rectangles.dat'
f2 = 'run_eff_perm.py'
f3 = 'octree_run_file.txt'

#  Iterate through each parameter to vary and generate multiple runs
num = 0
for i in range(6):
    lines_1 = []
    lines_1 = open_note(f1)
    nPoly = lines_1[9]
    nPoly_edited = nPoly.replace(nPoly[7:-1],param1[i])
    lines_1[9] = nPoly_edited
    for j in range(2):
        lines_2 = []
        lines_2 = open_note(f2)
        kf = lines_2[54]
```

```
        kf_edited = kf.replace(kf[12:17],param2[j])
        lines_2[54] = kf_edited

        lines_3 = []
        lines_3 = open_note(f3)

    num = num+1
    f_1 = 'gen_4_user_rectangles_' + str(num) + '.dat'
    f_2 = 'run_eff_perm_' + str(num) + '.py'
#         f_3 = 'octree_run_file_' + str(num) + '.txt'

    textfile_1 = open(f_1,'w')
    for element in lines_1:
        textfile_1.write(element)
    textfile_1.close()

    textfile_2 = open(f_2,'w')
    for element in lines_2:
        textfile_2.write(element)
    textfile_2.close()

    textfile_3 = open(f_3,'w')
    for element in lines_3:
        textfile_3.write(element)
    textfile_3.close()


# Loop for automating octree run files
num = 0
for i in range(12):
    num+=1
```

```
    lines_3 = []
    lines_3 = open_note(f3)
    line_1 = lines_3[0]
    line_1_edited = 'dfnGen
/dfnWorks/work/sim_3/L_6/gen_4_user_rectangles_'+str(num)+'.dat\n'
    lines_3[0] = line_1_edited

    line_2 = lines_3[1]
    line_2_edited = 'dfnFlow /dfnWorks/work/sim_3/L_6/UDFM_explicit.in\n'
    lines_3[1] = line_2_edited

    f_3 = 'octree_run_file_' + str(num) + '.txt'
    textfile_3 = open(f_3,'w')
    for element in lines_3:
        textfile_3.write(element)
    textfile_3.close()

# Iterate notes file === command line for running simulation
num = 0

a = 'python run_eff_perm'
b = '.py -name /dfnWorks/work/sim_3/L_6/run'
c = ' -input octree_run_file_'
d = '.txt -ncpu 10'

for i in range(12):
    num = num+1
    notes = a+str(num) + b+str(num) + c+str(num) +d
    print(notes)
```

Appendix E - Script for extracting fracture density (ρ)

```
import numpy as np
import pandas as pd
import h5py


class FractureDensity:

    """ This class created by Tolulope Agbaje on the 12th of December 2022 (A birthday gift for myself)
    The FractureDensity class takes 2 positional arguments namely permeability_cell and full_mesh
    which are the requisite data needed to extract volume-based fracture density


    permeability_cell = Cell data containing information for upscaled km and kf. permeability_cell is in .hf file format with (n,1) shape
    full_mesh = The full mesh data in .uge file format

    METHODS:
    1. The extract_mesh_permeability function extracts the permeability data
    2. The frac_den function extracts the cell volume (last column) from the full_mesh.uge file & calculate the volume-based fracture density

    """

    def __init__(self, permeability_cell, full_mesh):
        self.permeability_cell = permeability_cell
        self.full_mesh = full_mesh
```

```
    def extract_mesh_permeability(self):
        hf = h5py.File(self.permeability_cell, 'r')
        hf.keys()
        perm = hf.get('Permeability')
        perm_arr = np.array(perm)
        self.perm_df = pd.DataFrame(perm_arr) #Included self to be able to call in ot
        return self.perm_df

    def frac_den(self):
        import csv
        rows = []
        with open(self.full_mesh, 'r') as uge_file:
            reader = csv.reader(uge_file)
            for row in reader:
                row_val = row[0].split(' ')
                rows.append(float(row_val[-1]))
        self.fmesh = rows[1:]
        stop = int(rows[0])
        self.fm = pd.DataFrame({'mesh':self.fmesh[:stop]})
        perm = pd.concat([self.fm, self.perm_df], axis = 1)
        perm.rename(columns={0:'permeability'}, inplace = True)
        frac = perm.loc[perm['permeability'] != 1e-15]
        frac_den = np.sum(np.array(frac['mesh']))/np.sum(np.array(perm['mesh']))

        if len(self.fm) == len(self.perm_df):
            print('Full mesh and permeability are of equal dimension')
            return 'Fracture density:' + ' ' + str(frac_den), 'Full mesh size:' + ' ' + str(len(self.fm)),
'Permeability cell size:' + str(len(self.perm_df))
        else:
            print(f'Permeability size is {len(self.perm_df)} and full mesh size is {len(self.fm)}')
```

```
        print('Therefore,full mesh and permeability are not of equal dimension...please recheck your entry')



# An example of how to call the class
obj = FractureDensity('mesh_permeability.h5', 'full_mesh.uge'). # Instantiate an object of the class
perm_df = obj.extract_mesh_permeability() # Call the mesh_permeability method and assign to perm_df()
frac_density = obj.frac_den(). # Call the frac_den() method
frac_density
```