Journal of Petroleum Exploration and Production Technology (2021) 11:839–856
https://doi.org/10.1007/s13202-020-01079-w

ORIGINAL PAPER-EXPLORATION ENGINEERING

Developing a new algorithm for numerical modeling of discrete fracture network (DFN) for anisotropic rock and percolation properties

Erfan Hosseini[1] · **Mohammad Sarmadivaleh**[2] · **Zhongwei Chen**[3]

Received: 11 June 2020 / Accepted: 23 December 2020 / Published online: 22 January 2021

Abstract

The role of natural fractures in future reservoir performance is prominent. The fractured porous media is composed of an interconnected network of fractures and blocks of the porous medium where fractures occur in various scales and have a strong influence either when most of the flow is concentrated and them or when they act as barriers. A general numerical model for discrete fracture networks (DFN) is usually employed to handle the observed wide variety of fracture properties and the lack of direct fracture visualization. These models generally use fracture properties' stochastic distribution based on sparse and seismic data without any physical model constraint. Alternatively, a DFN model includes usual numerical geomechanical approaches like boundary element and finite element. But here, a geostatistical methodology has been used to generate a DFN model. In this paper, an alternative modeling technique is employed to create the realization of an anisotropic fractured rock using simulated annealing (SA) optimization algorithm. There is a notable positive correlation between fracture length and position. There are three principal subjects in a study of fractured rocks. Firstly, the network's connectivity, secondly, fluid flows through the system, and thirdly, dispersion. Here, connectivity of generated networks is considered. Continuum percolation is the mathematical model to study the geometry of connected components in a random subset of space. Different random realizations from the S.A. algorithm in four different sizes of $L = 100$, 150, 200, 250 at post-threshold condition are used as disordered media in percolation theory to compute percolation properties using Monte Carlo simulation. The percolation threshold (critical fracture density) and two crucial scaling exponents (β and υ) that dictate the model's connectivity behavior are estimated to over 200 realizations.

Keywords Discrete fracture networks (DFN) · Simulated annealing (S.A.) · Numerical modeling · Fluid flow · Monte Carlo simulation

✉ Erfan Hosseini
e.hosseini19@gmail.com

[1] Oil Industries Engineering and Construction Company (OIEC Group), Tehran, Iran

[2] Department of Petroleum Engineering, Curtin University, Perth, Australia

[3] School of Mechanical and Mining Engineering, Queensland University (UQ), Brisbane, Australia

Introduction

Fractures, as discontinuities within rocks in the crust, appear in vast length scales (Naderi et al. 2019). There are three principal models for fractured media: discrete, multi-continua, and hybrid models (Naderi et al. 2019). Discrete models are required when the continuum approach, where the whole system is modeled with an equivalent permeable medium characterized by permeability tensor, is not applicable. One of the most important criteria to choose a model is the scale of fractures concerning the study scale. For systems where the fracture scale is much less than the interest continuum approach's scale is sensible. Otherwise, the fracture network's geometry will be important, and discrete models would be considered (Kadkhodaei and Naderi 2017; Fathianpour et al. 2013). In most modeling of fractures and faults as part of heterogeneity models, fractures have been considered as discrete objects that have been given stochastic distributions often based on power-laws, calibrated either from the local core or seismic data or from more general correlations (Makel 2007; Darcel et al. 2003a, b; Heffer et al. 1999). One way of fracture modeling is the full geomechanical approach, i.e., modeling of the fracturing process, including the initiation of cracks, their propagation, coalescence, and mechanical interaction (Masihi et al. 2005; Dreuzy et al. 2004; Olson et al. 1998). Sahimi, Arbabi, and Tatomir used the minimization of an appropriate form of elastic energy along with geomechanics to model fractures (Tatomir 2007; Sahimi et al. 1993; Sahimi and Arbabi 1996). DFN has been successively used in recent years (Kang et al. 2019; Pichot et al. 2012; Berrone et al. 2013). Jing et al. (2017) developed a discrete fracture network for coal characterization. The model integrates the pore-scale roughness obtained from micro-computed tomography (micro-CT) imaging of coals into the fracture networks' discrete representation. Analysis of the fracture surfaces obtained from micro-CT imaging demonstrates random isotropic surfaces following a Gaussian distribution. The developed rough-walled discrete fracture network (RW-DFN) models are not restricted by the imaging resolution, so that they are favorable for direct numerical simulation of permeability. Besides, RW-DFN models can be constructed with an extended domain size to be incorporated into existing reservoir characterization frameworks to predict coal properties. A novel approach was presented by Song et al. (2018) to locally optimize DFN by integrating field tracer data based on an improved simulated annealing (SAW) process. Their approach has been validated with two field case examples by largely improving the performances of numerical simulation. The results of their work showed that optimization has a minimum effect on the original fracture density model. Additionally, the percentage of fractures updated in each perturbation of the proposed method decides the optimization's speed and precision level. The improved simulated annealing algorithm can also be widely used for other optimization problems. Jiang et al. (2019) investigated the effect of in situ stresses on fluid flow in a natural fracture network. Their simulations show that anisotropic stress loading tends to reduce fracture apertures and suppress fluid flow, resulting in decreased fractured rock's equivalent permeability. Anisotropic stresses may cause a significant sliding of fracture walls accompanied by shear-induced dilation and some preferentially oriented fractures, resulting in enhanced flow heterogeneity and channelization. In this study, a geostatistical approach is employed, which directly considers elastic energy in a fractured medium, a model that assumes all the fractures have existed, and the medium is in the equilibrium state. One main assumption is that elastic-free energy associated with fracture distribution follows a Boltzmann distribution. The simulated annealing (S.A.) algorithm is employed to minimize associated energy function. A mathematical model with which we can describe the connectivity and the percolation properties in complex geometries is called the percolation theory (Stauffer and Aharony 1992; Follin et al. 2014). The system is composed of objects distributed randomly and independently, which are not restricted to points on the fixed lattice (without any maximal concentration), such as fractured media, and continuum percolation gets rights into. Realizations of anisotropic random fracture networks produced by the simulated annealing algorithm will be used as a disordered system in the percolation theory's basic methodology. The critical properties of the resulting network will be estimated for different realizations for different system sizes.

Modeling

Fracture network modeling

In this paper, we use simulated annealing technique (Belayneh et al. 2006; Kirkpatrick et al. 1983) as an optimization tool with energy (objective) function to generate the fracture network's realization. Indeed, simulated annealing is a stochastic optimization technique that has been used in a variety of global optimization problems that involve objective functions with a large number of independent variables. Simulated annealing can be utilized with an energy function based on the covariance function to generate correlation in the system (Jing et al. 2017; Hamzehpour and Sahimi 2006). In the following adjustment and implementation of simulated annealing technique for this particular problem, an optimized configuration of fractures is outlined.

Simulated annealing algorithm

We can find an appropriate fracture configuration in 2D space with an anisotropic condition using simulated annealing. Application of simulated annealing needs to define several terms: (1) a set of possible structures of fractures and an initial design (2) a method for a small random symmetric change to the configuration (system perturbation method) (3) an objective function to be minimized (cost function) (4) an annealing schedule of changing a temperature-like parameter T, so that the system can reach its minimum global energy (5) stopping criterion.

Possible configurations and initial configuration Considering constraints of three characteristic parameters of fractures, i.e., position, length, and orientation, any configuration which obeys these constraints is possible. That is, fractures' centers must lie in the square of the side L with $x \in (0, L)$ and $y \in (0, L)$. There is no constraint for fractures' orientation, and fractures may have any direction. One way of minimizing the energy function defined by Eq. 1 is to reduce fractures' length.

$$E = \sum_{k=1}^{N} \sum_{\substack{k \neq l \\ l=1}}^{N} \left(A u_k u_l [\eta |\cos(\theta_k - \theta_l)| + |\cos(\alpha - \theta_l)\cos(\alpha - \theta_k)|] / r_{kl} \right) \tag{1}$$

where $A = E/16\pi\mu(1-\nu)$, N is the number of fractures in the system, and the components u_k and u_l are the displacements on fractures in k and l direction. α, θ_k, and θ_l specify the orientations of the distance vector r and fracture vectors u_k and u_l concerning the horizontal x-axis. This equation relates to the pairwise interactions of fractures to the total elastic energy of the system (Roy et al. 2010; Heffer and King 2005).

If simulated annealing procedure advances without any lower bound for fractures' length, fractures' size tends to zero. Two solutions have been proposed for this problem: (1) initializing model with large fractures (concerning square side) and defining a maturity control parameter and restriction of simulated annealing procedure using this parameter which is called accepted ratio (2) renormalizing fracture length distribution and keeping mean fracture length fixed. In this work, the first method is used.

One of the significant advantages of the simulated annealing technique is the independence of its initial configuration and the algorithm's convergence. As the fractures' initial design's choice does not affect the confluence, we can use different distributions for the fracture lengths and orientations in the algorithm. Here, Gaussian distribution for fracture length and uniform distribution for fracture orientation and position on (0, π) and (0, L) are used, respectively. A realization of the initial fracture network configuration is shown as follows (Fig. 1).

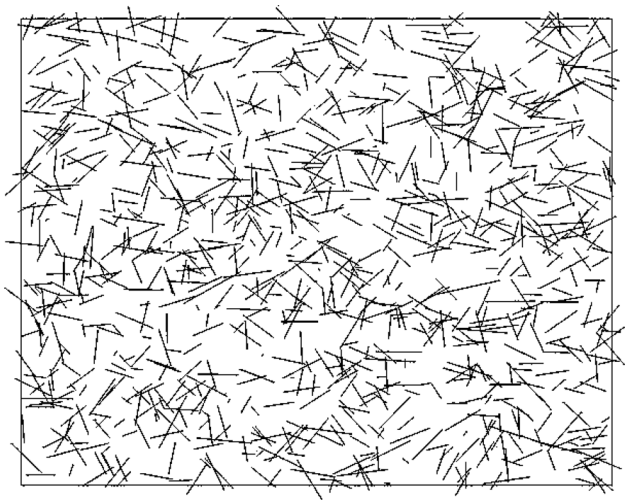

Fig. 1 Initial network configuration with Gaussian distribution for fracture length

System perturbation method After defining the initial configuration to progress to lower energy levels, a simulated annealing algorithm needs a new potential solution to evaluate. This new configuration must slightly differ from the prior accepted arrangement reached by the algorithm. The updating method for generating new near possible solutions from the currently accepted design (solution) may be fixed throughout, making the model mature or changed during algorithm progress. Contrary to the algorithm used by previous works, like, the updating method used to generate new potential solutions is not fixed and would change during algorithm progress. Model energy goes down, and the model becomes more mature, the number of further possible solution rejection will increase gradually. If the same schedule used for early iterations of the algorithm is used in late iterations, the number of overall rejection at the end of the process will be too high, and the model cannot reach a better configuration with lower energy. So a secondary schedule has been used to change the updating method. Thus, a more precise solution in more down run time is achieved.

To change the current solution and generate a new near potential solution, three characteristic properties of one randomly chosen model element (fracture) are altered slightly, i.e., length, orientation, and position. The updating equations must have the stuff to generate new potential solutions that differ somewhat from the current solution and be symmetric. Here, the Masihi and King (2007) equations are used. That is:

$$\begin{aligned}\theta_i^{\text{new}} &= \theta_i^{\text{old}} + 0.05\pi(2R-1)\\ l_i^{\text{new}} &= l_i^{\text{old}} + 0.05(2R-1)\\ rx_i^{\text{new}} &= rx_i^{\text{old}} + 0.5(2R-1)\\ ry_i^{\text{new}} &= ry_i^{\text{old}} + 0.5(2R-1)\end{aligned} \tag{2}$$

where R is the random number uniformly distributed on [0, 1]. The term $(2R-1)$ induces symmetry in simulated annealing algorithms. Fractures may get larger or smaller, rotate clockwise or counterclockwise, and move in the north, south, east, or west direction.

Objective function and transition So far, an initial configuration and an updating equation set have been presented. Now a decision-making criterion is required, so the model should transit to a new design or not. This criterion is the value or better change in the amount of some function called the objective function. Here, the objective function is the energy function presented in Eq. 1. To introduce anisotropy in this model, different η values have been assigned to x- and y-direction $\eta_y = 2$. Then, in the computation of energy between two fractures, an average of these values has been used. Averaging is based on length and orientation (projection of fractures on each axis). When a new configuration is generated by updating equations, the change in the energy function is computed. Suppose $\Delta E < 0$, the transition is accepted unconditionally. Otherwise, the new design is obtained or rejected based on the Metropolis–Hastings algorithm (with probability proportional to the Boltzmann's factor $e^{-\Delta E/T}$). This flexibility allows the algorithm to escape from local minima.

Temperature lowering schedule At any temperature, after a limited number of iterations system reaches a situation of maturity where the temperature parameter must be lowered. If the temperature parameter does not get reduced, the system maturity stays at that level captured into some loop, and algorithm progress would be too slow because of the high rate of acceptance for transitions with $\Delta E < 0$. To increase progress speed, this temperature should get lowered based on some appropriate schedule. This schedule directly depends on the size of the systems. Here, approximately after ten visits of each fracture in the system, the temperature gets lowered such that $T^{\text{new}} = kT^{\text{old}}$ and k ranges from 0.94 to 0.97. The following figure illustrates the variation in temperature against algorithm iteration. Here, colors indicate the energy levels of the system. The brighter the color, the higher the system energy level (Fig. 2).

Stopping criteria Depending on the particular system under study, time, computational limitations of available computers, and the study's purpose, different stopping criteria may be used. In this specific fracture system, stopping criteria may be defined as the average value ΔE for last N iterations, mean fractures' length, accepted, the total number of iterations, etc.

Here, a maturity control parameter is used to terminate algorithm progress. Indeed, this parameter is the acceptance ratio (A.R.) defined over the last 50,000 iterations if $AR < 0.01$, the algorithm would be released.

Periodic boundary condition (PBC)

The fractures in the model are distributed over a square of sides L. The limited lateral extension of this square will introduce boundary effects in the final configuration of fractures. This is because of the different stress experienced by fractures in the interior areas and the exterior areas.

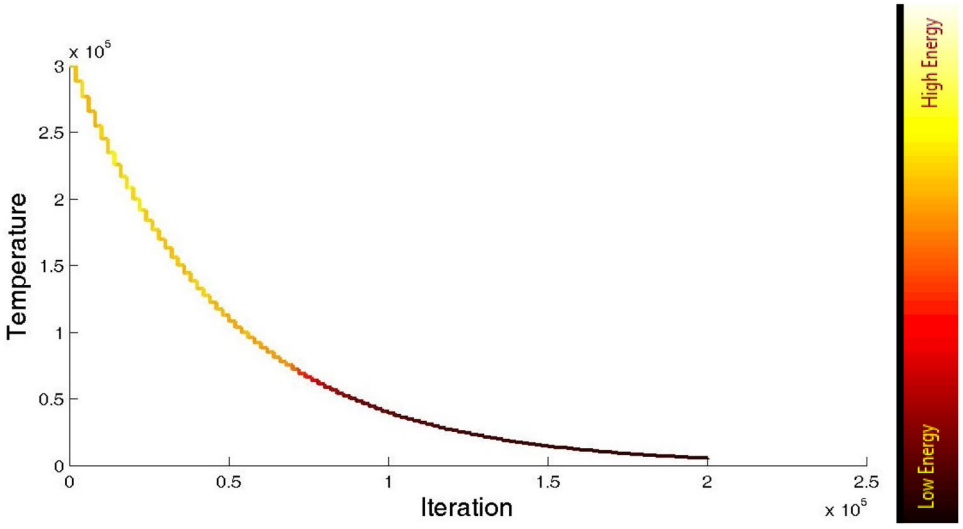

Fig. 2 Illustration of temperature lowering schedule ($T^{\text{new}} = kT^{\text{old}}$)

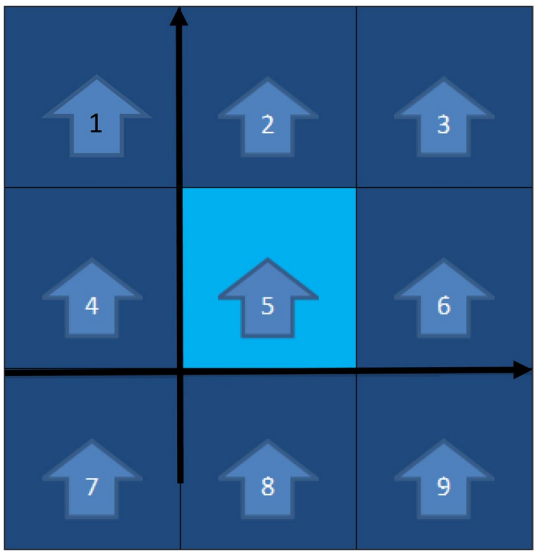

Fig. 3 Order and position of periodic boundary condition used in the algorithm in this work

By looking at equation one, this difference is noticeable. When a fracture is positioned on areas near boundaries, the average distance between this fracture and other fractures is smaller, and the contribution of this fracture in the total energy of the model will be smaller concerning other fractures in areas far from boundaries. Consider a simple boundary condition that replaces a fracture exiting from one side with exactly a similar fracture entering the other side. If this simple boundary condition that neglects the effect of boundaries is used, all the fractures would be distributed on the model's limitations, maximizing their distances from each other. This is because of the lack of any other fracture beyond the limits.

Masihi and King (2007) used a simple periodic boundary condition that considers the effect of boundaries using reflected imaginary fractures. In this boundary condition and the above simple one, if a fracture's end exits the square, keeping this fracture from the other side, an imaginary fracture exactly similar to leaving fracture with the same distance of boundary would be introduced in the court. But here, a different periodic boundary condition is used, which is physically more meaningful. The square is surrounded with eight exactly equal squares plus replacing exiting fractures from the other side. This boundary condition is shown as follows (Fig. 3).

MATLAB code for simulated annealing algorithm

In this work, the MATLAB programming language is used to write a simulated annealing algorithm. MATLAB (MATrix LABoratory) is a fourth-generation programming language and a software environment with various powerful computation and visualization tools in engineering projects. The primary data element is a matrix, so if you need a program that manipulates array-based data, it is generally fast to write and run in MATLAB, or vice versa. To speed up iterations of the simulated annealing algorithm's main loop, the center block's properties and eight surrounding blocks are unified in one matrix in a meaningful order such that there is only one length matrix for all fractures, whether in the center block or surrounding block. At least one-third of the whole time of this work has been spent writing code, especially in the percolation part. The first block of MATLAB codes for the simulated annealing algorithm, which initializes the model with 500 fractures, is given as follows. A full version of the MATLAB code of the algorithm is provided in the Appendix.

```
%This is the first lines of the main code that randomly initializes the model

clear;
rng('shuffle');                 %this command shuffles Random Number Generator
tic                             %keeps track of time
Clock


%************** orientation matrix *****************
Teta=zeros(1,4500);
Tetar = pi*rand(1,500);              %uniform distribution of orientation on [0,pi]

for i=1:500                          %unifies all Teta values in one matrix
   for j=1:9
      Teta(i+(j-1)*500)=Tetar(i);
   end
end

Tetai=Teta;                          %keeps initial Teta values

%******************* length matrix ******************
L=zeros(1,4500);
Lr =abs(random('norm',6,2,1,500));     %normal distribution of length  M=6, SD=2
for i=1:500
   for j=1:9                         %unifies length values in one matrix
      L(i+(j-1)*500)=Lr(i);
   end
end


Li=L;                                %keeps initial length matrix

%************* position matrices ******************
rx=zeros(1,4500);   ry=zeros(1,4500);
rxr = 100*rand(1,500);               %uniform distribution for position [0,100]
ryr=100*rand(1,500);

for i=1:500                          %unifies positions
   for j=0:2
      rx(i+3*j*500)=rxr(i)-100;
      rx(i+(3*j+1)*500)=rxr(i);
      rx(i+(3*j+2)*500)=rxr(i)+100;
   end
end

for i=1:500
   for j=1:3
      ry(i+(j-1)*500)=ryr(i)+100;
      ry(i+(j+2)*500)=ryr(i);
      ry(i+(j+5)*500)=ryr(i)-100;
   end
end
rxi=rx; ryi=ry;                      %keeps initial fracture position
```

Results and discussion

Generation of model realizations

Different realizations for different model sizes (or better fractures number) are required to investigate the fracture network's connectivity properties using percolation theory. In this work, totally 200 various realizations in four different system sizes of 100, 150, 200, and 250 using the simulated annealing algorithm have been generated. The final length distribution of fractures must be controlled between two extremes such that an investigation of connectivity is possible. Fractures in the final configuration must be in post-percolation threshold condition such that by randomly omitting fractures in the network, the particular percolation threshold can be extracted through the Monte Carlo technique. One extreme is when final fractures are too long, and the system is connected. The other height is when fractures are too short, and there is no spanning cluster of fractures.

Indeed, a balance between fracture number and fracture length in the final configuration must exist. So simulated annealing algorithm with various fractures number and initial length distribution (initial fracture density), perturbations method, initial temperatures and temperature lowering schedules, and stopping criteria have been examined through trial and error to reach an optimum combination of these significant factors. Different settings (algorithm input) used for simulated annealing algorithms in three different model sizes are given in Table 1.

Energy minimization during algorithm progress

As it was said before, by progressing the simulation process, the system's energy levels go down. This energy reduction process has two main phases. In the first phase, the model is at high energy levels, and the temperature is high. So the metropolis expression in the algorithm gives more chances for transitions, decreasing the system's total energy and having more opportunities to escape from local minima toward the global minimum. In other words, the hill-climbing property of the simulated annealing algorithm is more probable in this phase. Unlike the first phase, the system's second-phase energy levels are low, and the system reaches relative stability. In this phase, the system's opportunity to escape from traps is much shorter, and the system configuration only changes slightly. The following figure shows a typical behavior of system total energy against iteration.

These two phases of the total energy of the system look like two steps. In the upper stage, the system is too unstable, and there are severe fluctuations in the virtue of higher temperatures (brighter colors). Then, by lowering the temperature, system transits to a stable condition with lower energy levels and smooth changes because of relatively low temperature (darker colors).

The second useful graph is the graph of the energy difference between before and after perturbation. The two main phases of system energy here appear with a different characteristic. In this graph, fluctuations are evident in Fig. 4. But there is another factor in this graph which separates two phases. This factor is the exact symmetry condition around the x-axis. In the early iterations of the algorithm, there is a high acceptance rate for energy increasing transitions in virtue of the system's high temperature. So there is approximately asymmetry about the x-axis because nearly any change in the system's energy is accepted (brighter colors). As the number of iterations gets larger, the conditions change. By increasing the rate of rejection of energy increasing transitions, the graph becomes asymmetric around the x-axis and tends to the upper half with $DE > 0$ (darker colors) (Fig. 5).

The other useful graph is the cumulative number of accepted transitions against the cumulative number of iterations, which shows these two phases. The chart looks like a boomerang with a nearly straight wing and a gently curved wing. The left wing represents the first phase. This wing's slope is approximately 0.98, which means almost all perturbation has been accepted whether they have decreased energy or not. Then, the system transits to the second phase (curved region). Unlike a left wing, the right wing representing the second phase has a very gentle average slope of approximately 0.09. Nearly, only perturbations that reduce fracture length have been accepted (Fig. 6).

Table 1 Initial conditions of the simulated annealing algorithm

System size	No. of fractures	Mean	SD	T_i	Temp. schedule	No. of realizations	Stopping criterion
100	500	25	5	500,000	0.97	50	AR < 0.01
150	1150	25	5	600,000	0.96	50	AR < 0.01
200	2000	25	5	700,000	0.95	50	AR < 0.01
250	3150	25	5	900,000	0.94	50	AR < 0.01

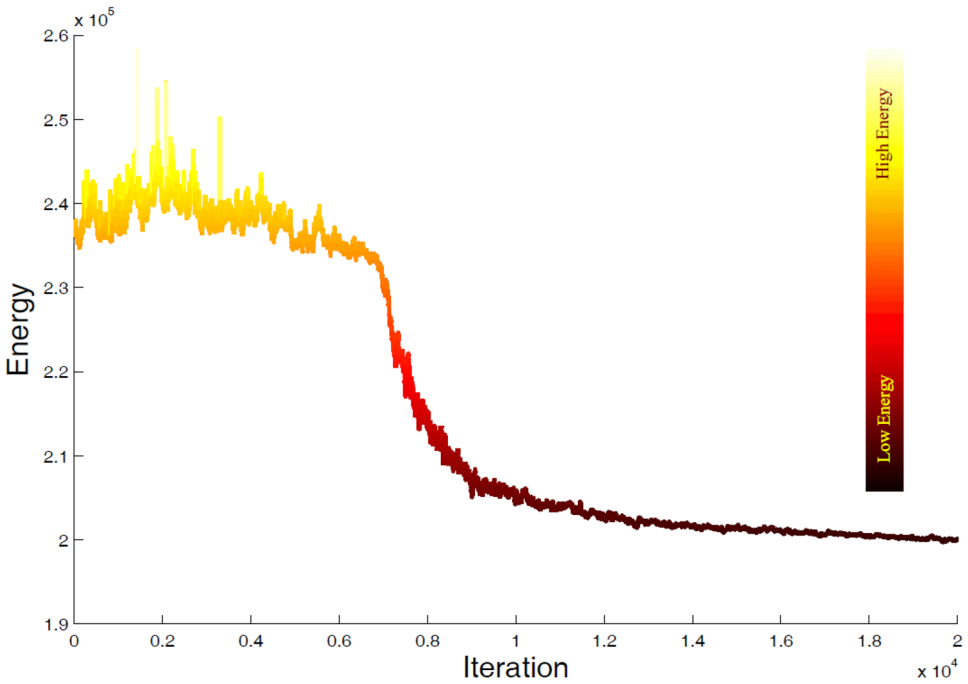

Fig. 4 Variation in total system energy (E) as a function of algorithm iteration (sampled with ratio 1:50)

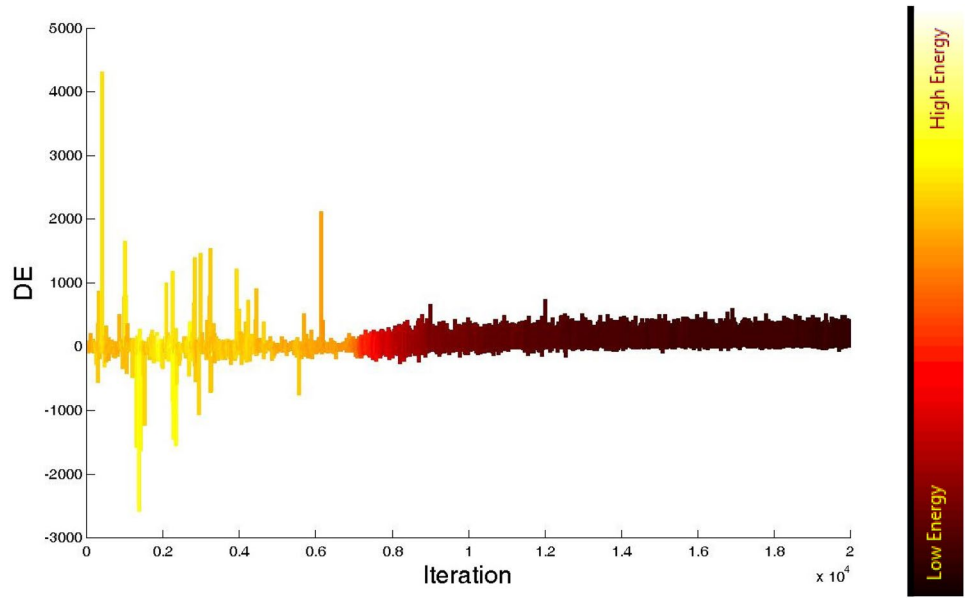

Fig. 5 Variation in energy difference before and after perturbation with iteration (1:50)

The exploitation of statistics of the model

In the following figure, one realization of a fracture network configuration has been given. The questions that come up are: How are different fracture network parameters distributed? Is there any correlation between the parameters of the final fracture configuration?

First of all, let inspect fracture length distribution. In Fig. 7, fracture size distribution has been presented. The fracture size distribution for this particular realization can be fitted either standard exponential or power-law distributions. As it has been mentioned before several times, different initial conditions (e.g., other fracture length distributions, different initial orientation) may be used for simulated annealing algorithm, and subsequently, additional final output can be generated. The fractures' absolute length values depend on initial values, and this model must be calibrated with local well log and core and seismic data. Figure 8 shows that the slope of fracture size distribution has been presented, which is a criterion of relative fracture size values.

Orientation is the other fundamental parameter of the final fracture network configuration. As shown in Fig. 8,

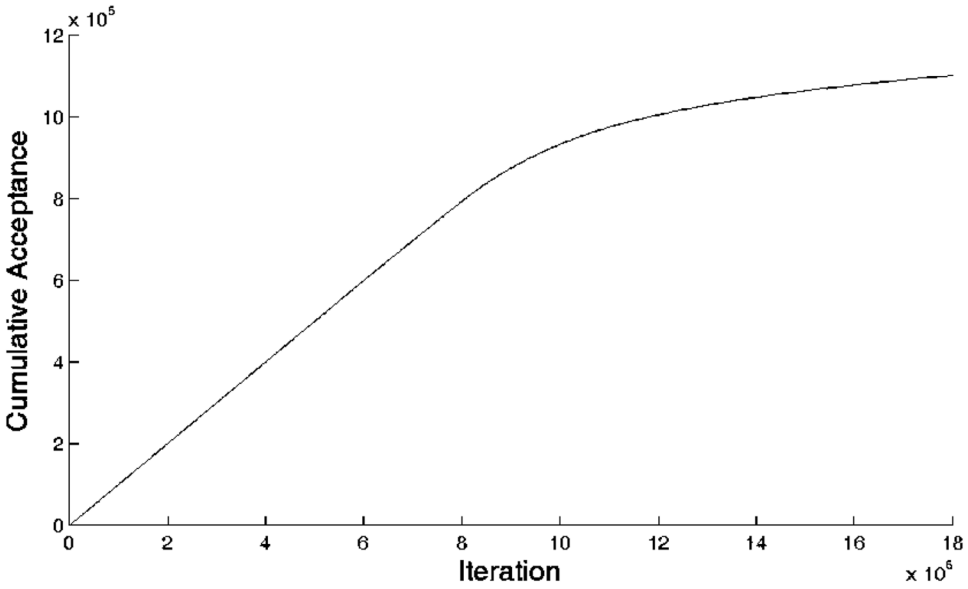

Fig. 6 Cumulative number of accepted transition against iteration

fractures generally constitute two fracture sets, a horizontal set and a vertical set. In the following figure, the distribution of orientation values of the fracture, the network has been given. In the upper part, distribution of all fracture sizes is offered, while relatively larger fractures have been shown. A diminution in all bars' size means smaller fractures are randomly distributed in the lower part, which is reasonable because smaller fractures have smaller contributions to the system's total energy. The metropolis term in the algorithm would have smaller values. This means smaller fractures have more chance to rotate randomly.

In the upper part of Fig. 9, the tall fracture orientation distribution on the model has been given. In this figure, two principal fracture sets have been shown as two modes of an asymmetric bimodal distribution, one more massive method in approximately 0 radians, the horizontal location, and the other in the approximately 1.6-radian vertical set. The latest height is nearly two times larger than the standing setting. The summary of the statistics of these two fracture sets is given as follows (Table 2).

Now, this question comes up: Is there any correlation between fracture length and fracture position? A particular type of correlation exists between these two parameters of the final fracture configuration. Bour and Davy (1999) and Lei et al. (2017) presented a new type of correlation in a fracture network between fracture length and fracture position. They defined parameter $d_c(l)$ to be the distance between the center of a fracture and its nearest neighbor. They showed there is a scaling law between this parameter and fractures length such that $d_c(l) \propto l^m$ where the exponent m varies in [0, 1]. The lower bound of this interval indicates there should be no correlation between fracture length and fracture position, while the upper bound corresponds to space-filling objects such as the apollonian described by Mandelbrot (1982) and Lei et al. (2016). Darcel et al. (2003a, b) and Hardebol et al. (2015), by using natural fracture network data, computed the exponent m from each fracture map within the range [0.1, 0.3].

It is straightforward for a fracture network that the initial condition of simulated annealing algorithm m will be nearly zero because the fractures were distributed randomly. But as the following figure shows, there is a correlation between fracture length and fracture position in the fractures' final configuration. There is a positive correlation between fracture length and placement of its nearest neighbor. This means, the larger the fracture, the larger the space around it (Fig. 10).

Here, summary statistics of all realizations have been given. In Table 3, important statistic parameters of all 200 realizations of fracture network such as relative number and length of two fracture sets, mean orientation angle for each fracture set separately, the slope of correlation between size and position of fractures have been computed.

Connectivity properties of the model

Eventually, in this section, the connectivity of this basic fracture model would be studied. Primarily, fracture network models are considered continuum percolation model cases with definitions of equivalent terms, and the percolation properties of these models are evaluated. Bour and Davy (1997) and Geiger (2019) modeled and quantified random fault networks' percolation properties following power-law fault length distribution, which differed from values for equal size fractures systems. Here, in this work,

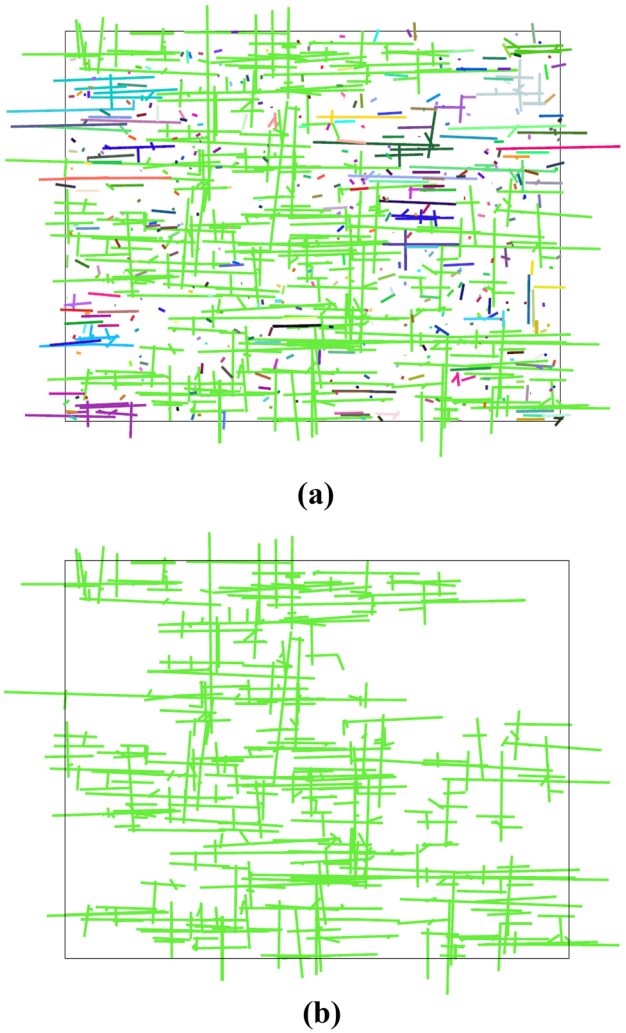

Fig. 7 **a** A final configuration of the fracture network with 1150 fractures and $L=150$. **b** Percolating cluster containing 453 fractures

the percolation threshold (p_c^{∞}), correlation length exponent (ν), and connectivity exponent (β) have been estimated.

Percolation threshold

From the practice point of view, all the systems under study are limited by size. As we know, percolation thresholds for these systems were defined as the occupancy where 50% of all realizations at that occupancy connect the system's boundaries (boundaries), and it was called the apparent percolation threshold. The apparent percolation threshold slightly varies with the system size due to finite size effects, which is modeled by:

$$p_c(L) - p_c^{\infty} \propto L^{-1/\nu} \tag{3}$$

The apparent percolation threshold p_c^{∞} is the actual percolation threshold (i.e., for infinitely large systems) and ν denotes the correlation length exponent (Song et al. 2018; Stauffer and Aharony 1992). This proportionality means the more massive the system size is, the better is the estimate of the actual percolation threshold. The error $\Delta(L)$ associated with apparent percolation values due to finite size effects also varies like $L^{-1/\nu}$ denotes the root mean square (RMS) deviation of apparent percolation threshold values for different realizations with the same system size. So without any knowledge ν, one may obtain an estimate of the percolation threshold by plotting $p_c(L)$ against $\Delta(L)$ by extrapolation $\Delta=0$. This method works particularly well for percolation, where it was introduced by Levinshtein et al. (1975). The equations and expressions above are the average percolation threshold by doing numerous Monte Carlo experiments for the same system size and check for when the spanning cluster appears for the first time when the system is getting filled slowly vice versa. In large networks, a standard method may be required. Here, an adapted version of the technique introduced by Stauffer and Aharony (1992) is used. First, check the connectivity for $N/2$ all fractures (randomly). If it does, decrease the number of fractures by $N/4$, otherwise, increase it by $N/4$. Now check again. If percolation occurs, reduce by $N/8$, otherwise increase it by $N/8$. Repeat this way to reach sufficient accuracy. After about ten such iterations, the onset of percolation is known with an accuracy sufficient for many purposes. This value is the threshold of this particular realization. By averaging over all realizations of this system size, one value $p_c(L)$ would be estimated (Fig. 11).

Correlation length exponent

There are three different ways to estimate the correlation length exponent. First is the scale dependency of the error associated with the apparent threshold. As mentioned in the previous section, it is sufficient to plot RMS deviation of apparent percolation threshold ($\Delta(L)$) for different system sizes in log–log scale paper. The slope of the resulting graph gives an estimate for the exponent ν. The alternative way is to use p_f and p_{1-f} as the occupancy probabilities at which the fraction f and $1-f$ of all realizations percolate, which is expected to have scale dependency given by Gawlinski and Stanley (1981) (Fig. 12),

$$p_{1-f} - p_f \propto L^{-1/\nu} \tag{4}$$

Different values f give different estimates, which can be averaged to obtain a more reliable estimate of ν. Finally, one can use equation five and take a log–log plot $\xi(p)$ against $p - p_c^{\infty}$ a relatively large system size.

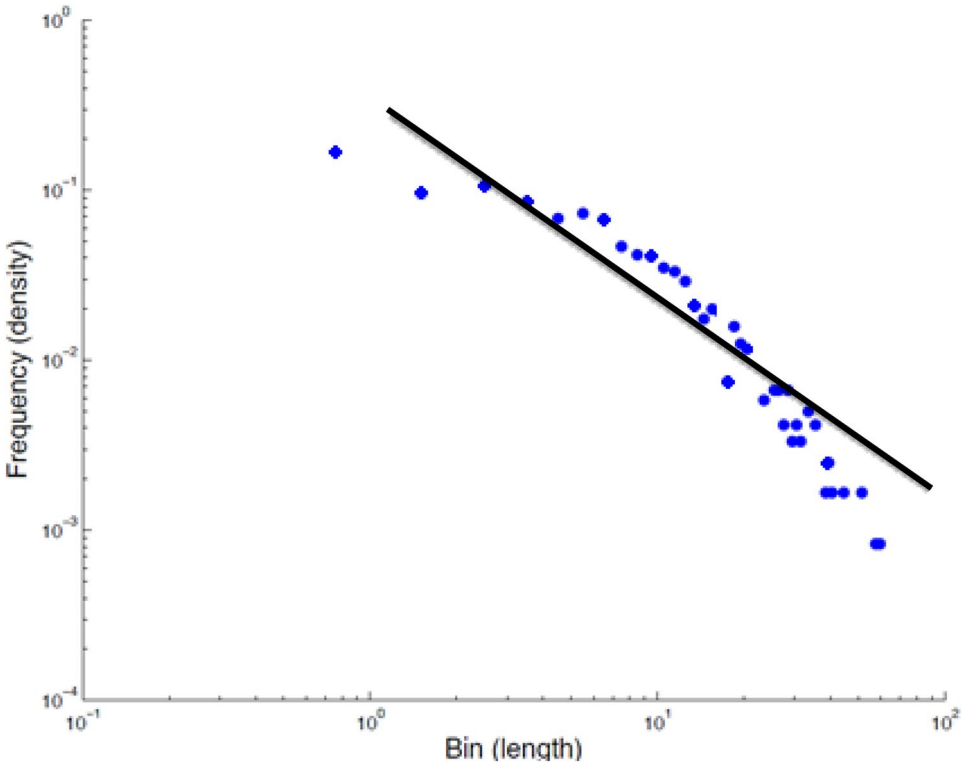

Fig. 8 Fracture length distribution of a typical realization. This distribution can be fitted with either exponential or power-law (slope ≈ −1.3)

$$\xi(p) \propto |p - p_c^{\infty}|^{-\nu} \tag{5}$$

Here, in this work, the first method has been used.

Connectivity exponent

There are two ways to obtain an estimate of the connectivity exponent (β). To estimate this exponent, one can take a log–log plot of $P(p, L)$ versus $p - p_c^{\infty}$ for a relatively large system size. The gradient of the produced curve would give an estimate of the connectivity exponent of the system. Alternatively, as the relevant scale is the system size at the apparent threshold, one can estimate β the percolation probability's finite size dependency (connected fraction), according to the following equation (Fig. 13).

$$P(p, L) \propto L^{-\beta/\nu} \tag{6}$$

Discussion

Heterogeneity in fractured petroleum reservoir rock is such an extent that it is impossible to model it fully. So, the significance of a statistical method is undeniable. Usually, different fracture network parameters such as size, orientation, and position are given statistical distributions and are calibrated using well log, seismic, and outcrop data of local reservoir rock. These models at least lack any physical constraint of the distribution of different characteristics of the network. In this work, a model was developed using a simulated annealing algorithm where it is the fundamental equation to be minimized was extracted from elasticity theory. Thus, the resulting model would have physical constraints rather than random unrestricted full statistical models. Section "The exploitation of statistics of the model" is devoted to the exploitation of statistics of the realizations of the model. This model is not capable of modeling the aperture of fractures. However, for example, to investigate the model's conductivity, one can use existing correlations between size and aperture of the fractures.

One of the most critical problems in the fractured reservoir is their high rate of uncertainty and variability due to different scale fractures, and this problem would appear in various stages of reservoir life from the early stages of drilling through field development and management. Fractures can act as hydraulic conductors or barriers to flow. The fracture network's connectivity is a crucial parameter in controlling fluid movement, which depends on individual fractures and the system's geometrical parameters.

One approach to investigating this extensively complicated network's connectivity is percolation theory as a basic mathematical model of a highly disordered media such as fracture networks. In Sect. "Fracture network modeling,"

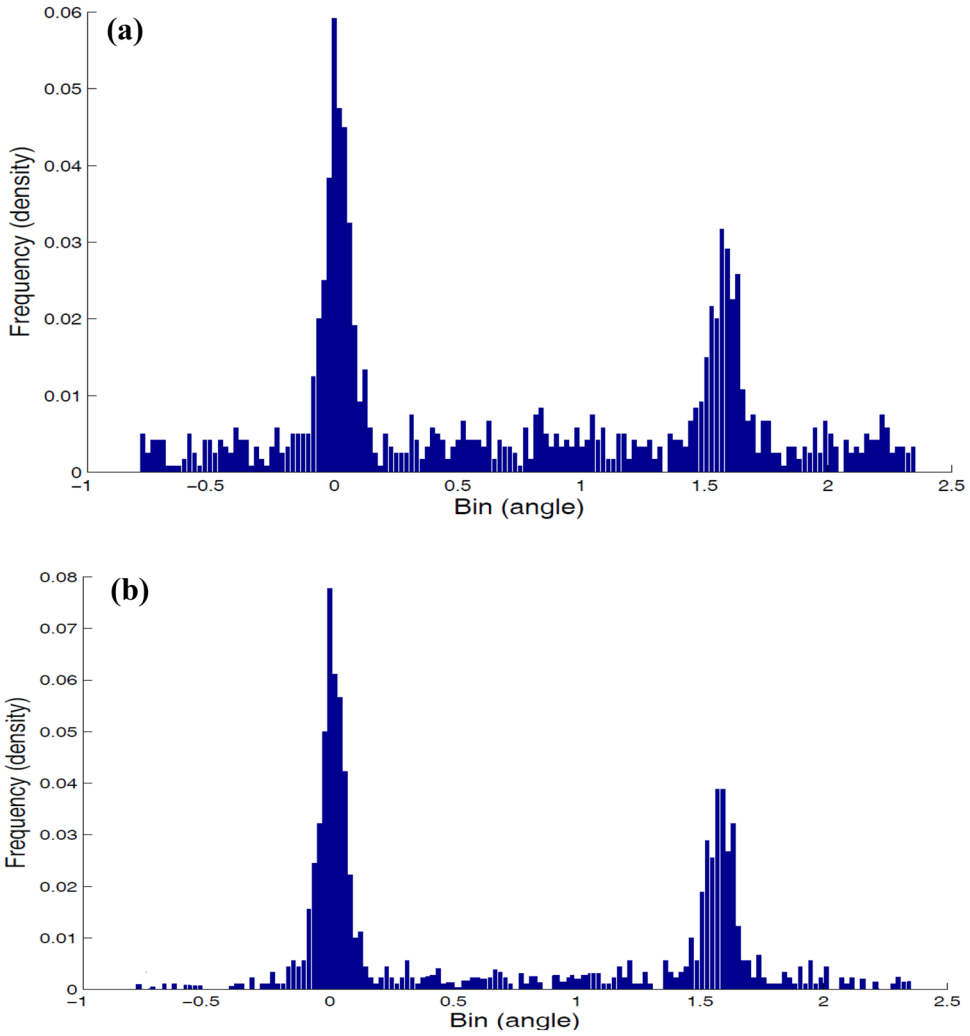

Fig. 9 Fracture orientation distribution on (above) all fractures (below) fractures larger than mean fracture length

Table 2 Summary statistics of two principal fracture sets

Set	No. of fractures	Length dist	Mean (L)	Orientation dist	Mean (θ)	SD (θ)
Horizontal	493	Exponential	7.083	Normal	0.010	0.31
Vertical	407	Exponential	4.271	Normal	1.563	0.37

three crucial percolation properties were estimated overall realizations. The percolation threshold (critical density) of the resulting system was evaluated. The scaling law exponents that appear in connectivity formulations and dictate the network's connectivity behavior were assessed. Here, a limited number (200) of realizations have been used. The results may be improved using more realizations.

Conclusions

Fracture petroleum reservoirs provide over 20% of reserve globally, and this number is nearly 90% in Iran. The usual approach in reservoir numerical modeling for fractured reservoirs is to use the statistical distribution of different fracture network parameters and calibrating to local well log, seismic, and outcrop data, and consequently, these models lack any physical restrictions. In this work, generalizations of a discrete fracture network for an anisotropic medium with different properties along each axis have been generated.

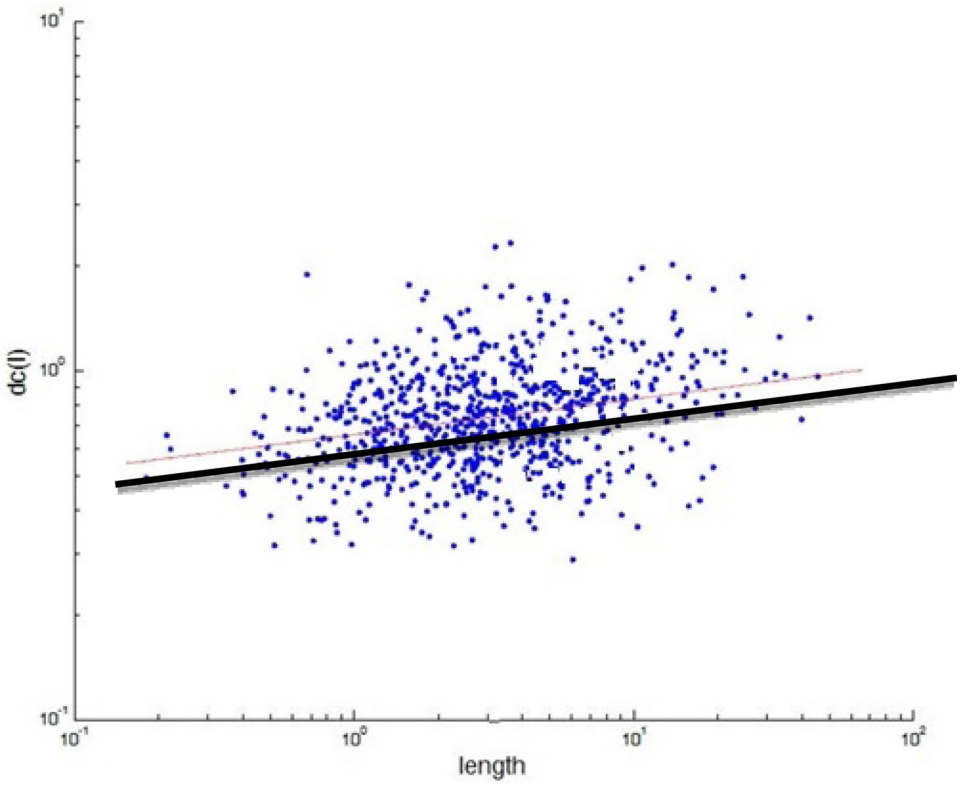

Fig. 10 Correlation between fracture length and position (slope ≈ 0.12)

Table 3 Summary statistics of all 200 realizations (M: Mean, SD: Standard deviation, R: Relative)

M (RN)	SD (RN)	M (M(RL))	SD (M(RL))	M (M(θ))	SD (M(θ))	M (slope)	SD (slope)
1.203	0.080	1.513	0.073	H: 0.004 V: 1.575	H: 0.013 V: 0.072	0.117	0.302

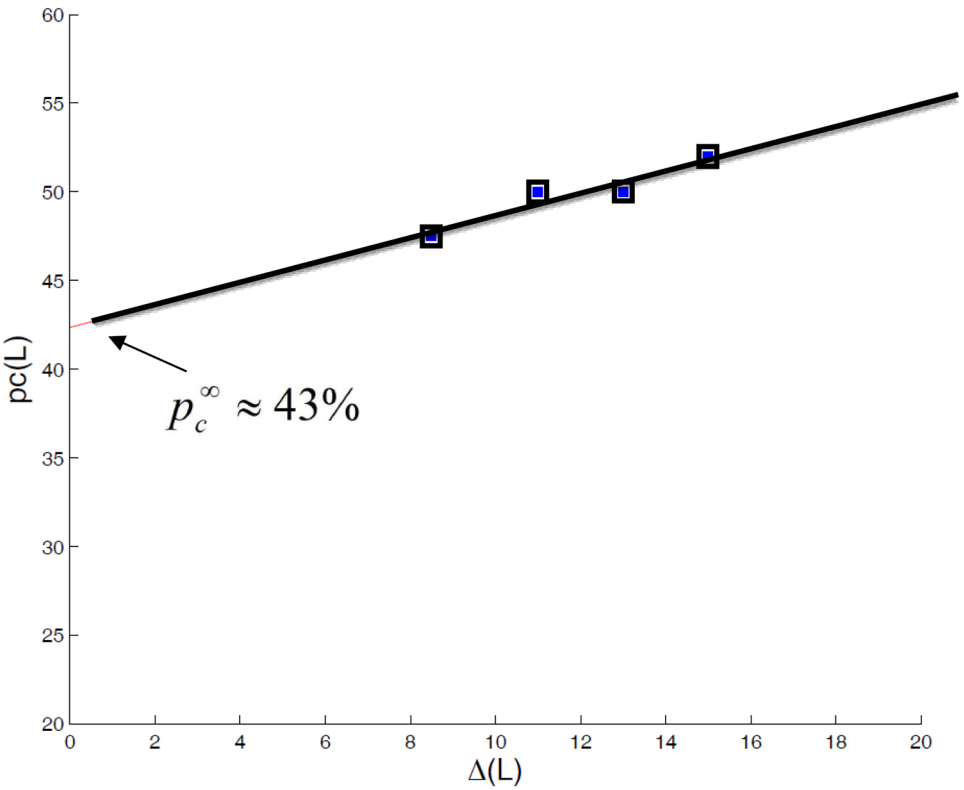

Fig. 11 Estimation of p_c^{∞} using extrapolation at $\Delta(L)=0$

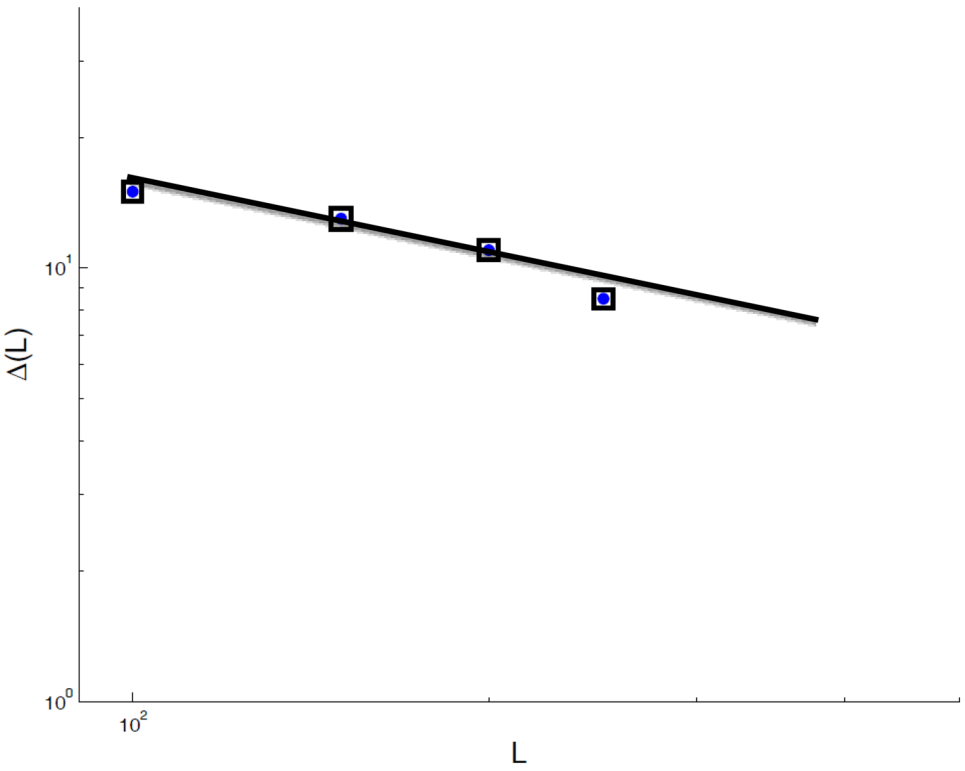

Fig. 12 Using the slope of variations $\Delta(L)$ against L in estimation of $1/\nu \approx 0.52 \pm 0.016$

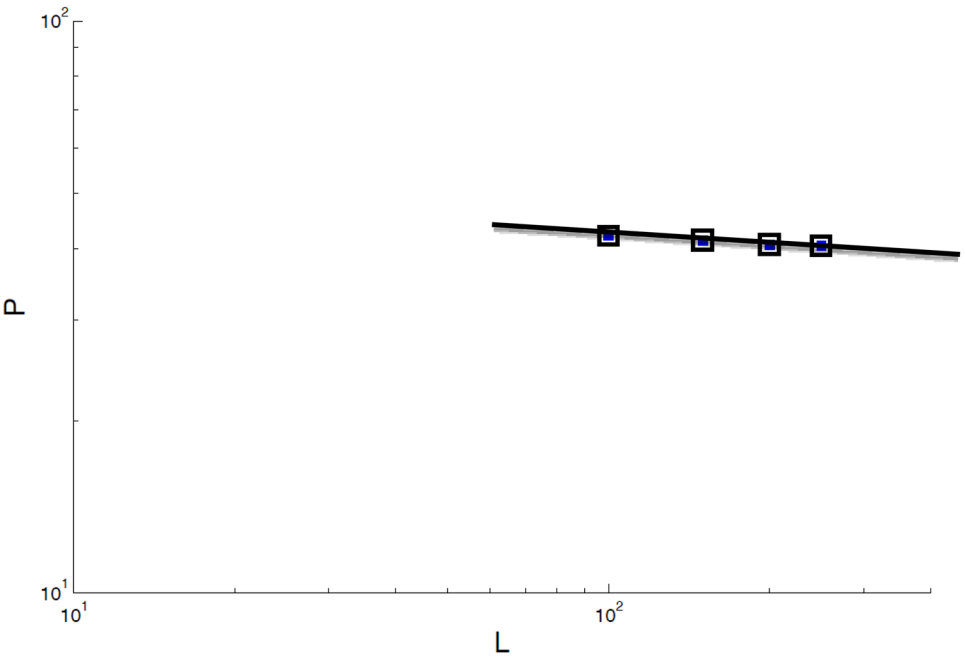

Fig. 13 Variation in P against L, $\beta/\nu \approx 0.05 \pm 0.017$

1. Fractures form two principal sets. Fractures along the x-axis are generally longer and more by number.
2. Fractures with length larger than fracture's mean length are more polarized along two principal axes. Smaller fractures due to their relatively smaller contribution to total system energy are nearly randomly distributed.
3. Log–log plot of fracture's length distributions has nearly linear form and can be fitted either power-law or harm-

ful exponential law. The mean slope of this line overall realizations is approximately −1.4.

4. There is a positive correlation between fracture's positions and lengths. Fractures with more extensive measurements are more robust to expel other fractures, and the nearest fracture would be more distance away.
5. There crucial parameter of the percolation model of the system has been estimated. These parameters were percolation threshold, correlation exponent, and connectivity exponent.
6. A new algorithm to identify fracture clusters has been developed (Appendix).

As future work, by reducing number of matrix which defines fracture network properties, for example, by uniting length, orientation, and position matrices, the computation elapsed time will decrease significantly. And, using aspect ratio as stopping criterion makes the output final fracture configuration be in same range by different properties such as orientation distribution and can be used in percolation methodology of connectivity.

Appendix: Matlab Codes

1-SA algorithm

```
rng('shuffle');
clear;
tic
clock

%**************  orientation matrix *****************
Teta=zeros(1,4500);
Tetar = pi*rand(1,500);

for i=1:500
   for j=1:9
      Teta(i+(j-1)*500)=Tetar(i);
   end
end

Tetai=Teta;

%******************  lenght matrix ******************
L=zeros(1,4500);
Lr =abs(random('norm',25,5,1,500));
for i=1:500
   for j=1:9
L(i+(j-1)*500)=Lr(i);
   end
end

Li=L;
%*************position matrices******************
rx=zeros(1,4500);   ry=zeros(1,4500);
rxr = 100*rand(1,500);
ryr=100*rand(1,500);

for i=1:500
  for j=0:2
      rx(i+3*j*500)=rxr(i)-100;
      rx(i+(3*j+1)*500)=rxr(i);
      rx(i+(3*j+2)*500)=rxr(i)+100;
   end
end

for i=1:500
   for j=1:3
      ry(i+(j-1)*500)=ryr(i)+100;
      ry(i+(j+2)*500)=ryr(i);
      ry(i+(j+5)*500)=ryr(i)-100;
   end
end

rxi=rx;
ryi=ry;

ux=zeros(1,4500);   uy=zeros(1,4500);

%*************fracture components*************
uxr= Lr .* cos(Tetar);
uyr= Lr.* sin(Tetar);
for i=1:500
   for j=1:9
   ux(i+(j-1)*500)=uxr(i);
   uy(i+(j-1)*500)=uyr(i);
   end
end


uxi=ux;
uyi=uy;
A=1;
K=0;
E=zeros(1,1000000);
Eti=0;
E1=0;
%******************initial energy*******************

for i=2001:2500
   for j=1:4500
      if (j<2001) || (j>2500)
         rij=[rx(i)-rx(j);ry(i)-ry(j)];

E1=energyfun(ux(i),ux(j),uy(i),uy(j),rij(1),rij(2),4*(sin(Teta(i)))^2+2*(cos(Teta(i)))^2,4*(sin(Teta(j)))^2+2*(cos(Teta(j)))^2);
         Eti=Eti+E1;
      elseif i~=j
         rij=[rx(i)-rx(j);ry(i)-ry(j)];

E1=energyfun(ux(i),ux(j),uy(i),uy(j),rij(1),rij(2),4*(sin(Teta(i)))^2+2*(cos(Teta(i)))^2,4*(sin(Teta(j)))^2+2*(cos(Teta(j)))^2);
```

```
        Eti=Eti+E1/2;
      end
    end
end


E(1)=Eti;
disp(' ');
disp(['                                                    Initial energy :',num2str(E(1))]);

%*****************SA*************************

NT=0;
na=0;                                                  AR=1;
Temp=500000;
DEminuscount=0;
Metgreaterone=0;
Qcount=0;
PEb=zeros(1,1000000);
PEa=zeros(1,1000000);
DE=zeros(1,1000000);
NA=zeros(1,1000000);
RAR=1;
progmet=50000;
Met=0;

while (NT<1000000) && (RAR>0.01)

    n = unidrnd(500);
    R1=rand;  R2=rand;   R3=rand;   R4=rand;

    TetaN=Teta(n+2000)+0.05*pi*(2*R1-1) ;     LN=abs(L(n+2000)+0.1*(2*R2-1)) ;     UXN=LN*cos(TetaN)  ;
UYN=LN*sin(TetaN)  ;

    RXN=zeros(1,9);

    if   rx(n+2000)+0.5*(2*R3-1)> 100
       RXNm=rx(n+2000)+0.5*(2*R3-1)-100;
       for i=1:3
          RXN(1+(i-1)*3)=RXNm-100;
          RXN(2+(i-1)*3)=RXNm;
          RXN(3+(i-1)*3)=RXNm+100;
       end

    elseif   rx(n+2000)+0.5*(2*R3-1)<0
       RXNm=rx(n+2000)+0.5*(2*R3-1)+100;
       for i=1:3
          RXN(1+(i-1)*3)=RXNm-100;
          RXN(2+(i-1)*3)=RXNm;
          RXN(3+(i-1)*3)=RXNm+100;
       end
    else
       RXNm=rx(n+2000)+0.5*(2*R3-1);

       for i=1:3
          RXN(1+(i-1)*3)=RXNm-100;
          RXN(2+(i-1)*3)=RXNm;
          RXN(3+(i-1)*3)=RXNm+100;
       end
    end

    RYN=zeros(1,9);

    if   ry(n+2000)+0.5*(2*R4-1)> 100
       RYNm=ry(n+2000)+0.5*(2*R4-1)-100;
       for i=1:3
          RYN(i)=RYNm+100;
          RYN(i+3)=RYNm;
          RYN(i+6)=RYNm-100;
       end

    elseif  ry(n+2000)+0.5*(2*R4-1)<0
       RYNm=100+ry(n+2000)+0.5*(2*R4-1);
       for i=1:3
          RYN(i)=RYNm+100;
          RYN(i+3)=RYNm;
          RYN(i+6)=RYNm-100;
       end
    else
       RYNm=ry(n+2000)+0.5*(2*R4-1);
       for i=1:3
          RYN(i)=RYNm+100;
          RYN(i+3)=RYNm;
          RYN(i+6)=RYNm-100;
       end
    end



    Ep1=0;
    Ep21=0;
    Ep22=0;
    E1=0;
    E21=0;
    E22=0;

    %*************before********************
    for i=1:4500
       if (i~=n+2000)

          r1=[rx(n+2000)-rx(i);ry(n+2000)-ry(i)];

E1=energyfun(ux(n+2000),ux(i),uy(n+2000),uy(i),r1(1),r1(2),4*(sin(Teta(n+2000)))^2+2*(cos(Teta(n+2000)))^
2,4*(sin(Teta(i)))^2+2*(cos(Teta(i)))^2);
          Ep1=Ep1+E1;
       end
    end

    PEb(NT+1)=Ep1;

    %**************after********************
    for i=1:9
       for j=1:500
          if j~=n
             k=(i-1)*500+j;
             r21=[RXNm-rx(k);RYNm-ry(k)];

E21=energyfun(UXN,ux(k),UYN,uy(k),r21(1),r21(2),4*(sin(TetaN))^2+2*(cos(TetaN))^2,4*(sin(Teta(j)))^2+2
*(cos(Teta(j)))^2);
```

```
            Ep21=Ep21+E21;
        end
    end
end

for i=1:9
    if i~=5
        r22=[RXN(i)-RXNm;RYN(i)-RYNm];

E22=energyfun(UXN,UXN,UYN,UYN,r22(1),r22(2),4*(sin(TetaN))^2+2*(cos(TetaN))^2,4*(sin(TetaN))^2+2
*(cos(TetaN))^2);
        Ep22=Ep22+E22;
    end
end

PEa(NT+1)=Ep21+Ep22;


%**************** Transition : Yes or No *****************

DE(NT+1)=PEa(NT+1)-PEb(NT+1);
Met=exp(-DE(NT+1)/(Temp*(0.96^(floor(na/5000)))));

if DE(NT+1)<0
    for i=1:9
        Teta(n+(i-1)*500)=TetaN; L(n+(i-1)*500)=LN; ux(n+(i-1)*500)=UXN; uy(n+(i-1)*500)=UYN;
rx(n+(i-1)*500)=RXN(i); ry(n+(i-1)*500)=RYN(i);
    end
    na=na+1;
    E(NT+2)=E(NT+1)+DE(NT+1);
    DEminuscount=DEminuscount+1;
    % Met was here
elseif Met >=1
    for i=1:9
        Teta(n+(i-1)*500)=TetaN; L(n+(i-1)*500)=LN; ux(n+(i-1)*500)=UXN; uy(n+(i-1)*500)=UYN;
rx(n+(i-1)*500)=RXN(i); ry(n+(i-1)*500)=RYN(i);
    end
    na=na+1;
    E(NT+2)=E(NT+1)+DE(NT+1);
    Metgreaterone=Metgreaterone+1;
else
    Q=rand;
    if Q<=Met
        for i=1:9
            Teta(n+(i-1)*500)=TetaN; L(n+(i-1)*500)=LN; ux(n+(i-1)*500)=UXN; uy(n+(i-1)*500)=UYN;
rx(n+(i-1)*500)=RXN(i); ry(n+(i-1)*500)=RYN(i);
        end
        na=na+1;
        E(NT+2)=E(NT+1)+DE(NT+1);
        Qcount=Qcount+1;
    else
        E(NT+2)=E(NT+1);
    end
end

NT= NT+1;
AR= (na+1)/NT;
NA(NT)=na;

if floor(NT/progmet)-floor((NT-1)/progmet)>0
    D=floor(NT/progmet)*progmet;
    clock
    disp(' ');
    disp([' Now NT= ' ,num2str(D)]);
    RAR=(NA(NT)-NA(NT-10000))/10000;
end
End
```

2- Cluster identification script (Algorithm Developed by Authors)

```
ClustNum=zeros(1,N);
ClustNum(1)=1;
fracint=zeros(1,N);

for i=1:N
    ttt=0;
    zz=60000*ones(1,N);
    if ClustNum(i)==0
        zz(i)=max(ClustNum)+1;
        ClustNum(i)=max(ClustNum)+1;
        for j=1:N
            if j~=i
                Rij=sqrt((rx(i)-rx(j))^2+(ry(i)-ry(j))^2);
                alpha=atan((ry(i)-ry(j))/(rx(i)-rx(j)));
                Ai=Rij*abs(sin(alpha-Teta(j))/sin(Teta(i)-Teta(j)));
                Aj=Rij*abs(sin(alpha-Teta(i))/sin(Teta(i)-Teta(j)));
                if (Ai<=L(i)/2) && (Aj<=L(j)/2)
                    ttt=ttt+1;
                    if  ClustNum(j)==0
                        ClustNum(j)=ClustNum(i);
                        zz(j)=ClustNum(i);
                    elseif ClustNum(j)<=ClustNum(i)
                        ClustNum(i)=ClustNum(j);
                        for w=1:N
                            if ClustNum(w)==ClustNum(j)
                                zz(w)=ClustNum(j);
                            end
                        end
                    else
                        for x=1:N
                            if ClustNum(x)==ClustNum(j)
                                zz(x)=ClustNum(j);
                            end
                        end
                        ClustNum(j)=ClustNum(i);
                    end
                end
            end
        end
        fracint(i)=ttt;
        for s=1:N
            if zz(s)<60000
                ClustNum(s)=min(zz);
            end
        end


    elseif i==1
        for j=1:N
```

```
        if j~=i
            Rij=sqrt((rx(i)-rx(j))^2+(ry(i)-ry(j))^2);
            alpha=atan((ry(i)-ry(j))/(rx(i)-rx(j)));
            Ai=Rij*abs(sin(alpha-Teta(j))/sin(Teta(i)-Teta(j)));
            Aj=Rij*abs(sin(alpha-Teta(i))/sin(Teta(i)-Teta(j)));
            if (Ai<=L(i)/2) && (Aj<=L(j)/2)
                ttt=ttt+1;
                ClustNum(j)=1;
            end
        end
    end
    fracint(1)=ttt;
else
    for a=1:N
        if ClustNum(a)==ClustNum(i)
            zz(a)=ClustNum(i);
        end
    end
    for j=1:N
        if j~=i
            Rij=sqrt((rx(i)-rx(j))^2+(ry(i)-ry(j))^2);
            alpha=atan((ry(i)-ry(j))/(rx(i)-rx(j)));
            Ai=Rij*abs(sin(alpha-Teta(j))/sin(Teta(i)-Teta(j)));
            Aj=Rij*abs(sin(alpha-Teta(i))/sin(Teta(i)-Teta(j)));
            if (Ai<=L(i)/2) && (Aj<=L(j)/2)
                ttt=ttt+1;
                if ClustNum(j)==0
                    ClustNum(j)=ClustNum(i);
                    zz(j)=ClustNum(i);
                elseif ClustNum(j)<=ClustNum(i)
                    ClustNum(i)=ClustNum(j);

                    for p=1:N
                        if ClustNum(p)==ClustNum(j)
                            zz(p)=ClustNum(j);
                        end
                    end

                else
                    for q=1:N
                        if ClustNum(q)==ClustNum(j)
                            ClustNum(q)=ClustNum(i);
                            zz(q)=ClustNum(i);
                        end
                    end
                    ClustNum(j)=ClustNum(i);
                end
            end
        end
    end
    fracint(i)=ttt;
    for s=1:N
        if zz(s)<60000
            ClustNum(s)=min(zz);
        end
    end
end
end
```

```
u=0;
for i=1:max(ClustNum)
    z=0;
    for j=1:N
        if ClustNum(j)==i
            z=z+1;
        end
    end
    if z>0
        u=u+1;
        ClustName(u)=i;
        ClustSize(u)=z;
    end
end
numberofcluster=length(ClustName);
```


References

Belayneh M, Masihi M, Matthäi SK, King PR (2006) Prediction of vein connectivity using the percolation approach: model test with field data. J Geophys Eng 3(3):219–229

Berrone S, Pieraccini S, Scialo S (2013) A PDE-constrained optimization formulation for discrete fracture network flows. SIAM J Sci Comput. https://doi.org/10.1137/120865884

Bour O, Davy P (1999) Clustering and size distributions of fault pattern: theory and measurements. Geophys Res Lett 26(13):2001–2004

Darcel C, Bour O, Davy P, de Dreuzy JR (2003) Connectivity properties of two-dimensional fracture networks with stochastic fractal correlation. Water Resour Res

Darcel C, Bour O, Davy P (2003) Cross-correlation between length and position in real fracture networks. Geophys Res Lett.

Dreuzy JR, Darcel C, Davy P, Bour O (2004) Influence of spatial correlation of fracture centers on the permeability of two dimensional fracture networks following a power law length distribution. Water Resour Res

Fathianpour N, Mokhtari AR, Naderi H (2013) Improving estimation of STOIIP in one of Iranian South oil field using porosity pattern simulation through filtersim algorithm. In: Proceedings of the 4th Iranian mining engineering conference, mining engineering faculty, University of Tehran.

Follin S, Hartley L, Rhén I et al (2014) A methodology to constrain the parameters of a hydrogeological discrete fracture network model for sparsely fractured crystalline rock, exemplified by data from the proposed high-level nuclear waste repository site at Forsmark. Sweden Hydrogeol J 22(2):313–331

Hardebol NJ, Maier C, Nick H, Geiger S, Bertotti G, Boro H (2015) Multiscale fracture network characterization and impact on flow: a case study on the Latemar carbonate platform. J Geophys Res Solid Earth 120(12):8197–8222

Heffer KJ, King PR, Jones ADW (1999) Fracturing modelling as part of integrated reservoir characterization. In: SPE paper, Middle East Oil Show and Conference, 20–23 February 1999, Bahrain.

Jiang C, Wang X, Sun Z, Lei Q (2019) The role of in situ stress in organizing flow pathways in natural fracture networks at the percolation threshold. Geofluids 2019:1–14

Jing Y, Armstrong RT, Mostaghimi P (2017) Rough-walled discrete fracture network modelling for coal characterization. Fuel 191:442–453

Kadkhodaei A, Naderi H (2017) Investigation on effect of natural fractures on experienced breakdown pressures in comparison with expected pressures based on geomechanic study in one of Iranian south oilfields. In: Proceedings of the 2nd national conference on petroleum geomechanics.

Kang PK, Lei Q, Dentz M, Juanes R (2019) Stress-induced anomalous transport in natural fracture networks. Water Resour Res 55(5):4163–4185

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680

Lei Q, Latham JP, Xiang J (2016) Implementation of an Empirical Joint Constitutive Model into Finite-Discrete Element Analysis of the Geomechanical Behaviour of Fractured Rocks. Rock Mech Rock Eng 49(12):4799–4816

Lei Q, Latham JP, Tsang CF (2017) The use of discrete fracture networks for modeling coupled geomechanical and hydrological behavior of fractured rocks. Comput Geotech 85:151–176

Mäkel GH (2007) The modeling of fractured reservoirs: constraints and potential for fracture network geometry and hydraulics analysis. Geol Soc 292(1):375–403

Masihi M, King P (2007) Connectivity of spatially correlated fractures:simulation and ield studies. In: Paper SPE 107132 presented at the SPE EUROPEC/EAGE Conference and Exhibition, London

Masihi M, King PR, Nurafza P (2005) Fast estimation of performance in fractured reservoirs using percolation theory. In: SPE Paper.

Naderi H, Fathianpour N, Tabaei M (2019) MORPHSIM: a new multiple-point pattern-based unconditional simulation algorithm using morphological image processing tools. J Pet Sci Eng 173:1417–1437. https://doi.org/10.1016/j.petrol.2018.09.028

Song S, Hou J, Sun S, Li Y, Wang X, Dou L, Liu Y, Kang Q, Huang S (2018) Local optimization of DFN by integrating tracer data based on improved simulated annealing. J Pet Sci Eng 170:858–872

Stauffer D, Aharony A (1992) Introduction to percolation theory. Taylor and Francis, London, p 181

Tatomir AB (2007) Numerical investigations of flow through fractured porous media. In: Master Thesis.