Contents lists available at ScienceDirect

Advances in Water Resources

journal homepage: www.elsevier.com/locate/advwatres

Highlights

Stochastic inversion of discrete fracture networks using genetic algorithms

Advances in Water Resources xxx (xxxx) xxx

Fleford Redoloza, Liangping Li*, Arden Davis

- Discrete Fracture Networks coupling with PFLOTRAN were used to simulate flow and transport.
- Genetic algorithms are used to inversely calibrate fractures.
- Synthetic example is used to demonstrate the effectiveness of inversion.

Contents lists available at ScienceDirect

Advances in Water Resources

journal homepage: www.elsevier.com/locate/advwatres

Stochastic inversion of discrete fracture networks using genetic algorithms

Fleford Redoloza, Liangping Li *, Arden Davis

Department of Geology and Geological Engineering, South Dakota School of Mines and Technology, Rapid City, 57701, USA

ARTICLE INFO

Keywords:
Discrete fracture networks
Genetic algorithms
Stochastic inversion
Subsurface flow and transport

ABSTRACT

Reservoir management and contaminant transport simulations rely on accurate modeling of the subsurface. This task becomes more challenging when the reservoir of interest also has a large amount of fractures. Inverse modeling of these fractured media involves first performing multiple field tests such as pumping tests, tracer tests, or seismic surveys. Inverse modeling methods then are used to find potential geologic models that produce simulated results that match field observations. Popular methods include the Ensemble Kalman Filter, Markov Chain Monte Carlo, and simulated annealing. A common challenge these methods face is that inverse modeling of fractured media is inherently a multiobjective optimization task. The inverse modeling method must find a discrete fracture model that produces the same flow characteristics as observed in field pumping or tracer tests. But it also must find a discrete fracture model with a fracture network that matches the fracture parameter distribution observed by the field measurements such as seismic surveys. This challenge can be approached in two steps. The first step is producing discrete fracture network models with parameter distributions that match field observations from seismic surveys. This study focused on the second step, involving the development of a method that can take a population of discrete fracture networks and generate new fracture networks in a way that preserves the fracture parameter distribution. This was done using a genetic algorithm modified to apply to the domain of discrete fracture networks. During genetic mixing, the fractures of the child model are generated by randomly copying over fractures from the parent models. This process ensures the child model adopts the fracture parameter distribution of the parent models. This study tests the effectiveness of this genetic algorithm on a synthetic example. The results of the experiment show the genetic algorithm is able to effectively produce a population of discrete fracture models with breakthrough curves that match the curves of the reference model.

1. Introduction

Fracture network models are important tools across multiple industries. The most prominent use of fracture network models is in the petroleum industry where they are used for determining the optimal methods for fracturing and recovering petroleum from reservoirs. Another application for fracture network models is geothermal energy production. To design effective methods for producing fractures that transfer thermal energy from the crust to the fluids flowing through the fracture network, researchers rely on accurate fracture network models. A wide variety of conceptual models exist for simulating flows through fractured rock. Concepts include modeling the rock is a discrete network of fractures, another as nonuniform continuum of hydraulic parameters, or even as a hybrid where the rock is modeled as a nonuniform continuum that contains a small number of dominant discrete features (Neuman, 2005). Such variety exists because of the different geologic contexts that produces a range of flow behavior. This study focuses on the use discrete fracture network models to simulate flows through fractured rock. This conceptual model would be suitable to simulating flows through impermeable crystalline rock. Unlike traditional groundwater flow simulations that discretize the domain as a grid of cells each with its hydraulic properties, discrete fracture networks describe fractures as a set of finite planes with a defined shapes, sizes, locations and orientations. The fracture parameters often are defined on the basis of a probability distribution that imitates distributions of fractures measured in the field. During flow simulations, the fluids strictly flow through these fracture planes with the flow behavior governed by the hydraulic properties of the fracture planes and how the fracture planes are connected. Note that for real fractured rock, fluid exchange can exist between the fracture and the matrix. For this work, we assume that most of the fluid dynamics is governed by the geometry of fractures and how they are connected to each other. The goal is to investigate how the complexities of geometry-dominated flow affect the performance of a proposed method using genetic algorithms. Thanks to the flexibility of genetic algorithms, the

* Corresponding author.
E-mail address: Liangping.Li@sdsmt.edu (L. Li).

https://doi.org/10.1016/j.advwatres.2023.104477
Received 17 February 2023; Received in revised form 11 May 2023; Accepted 2 June 2023

proposed method can be adjusted to handle scenarios where the fluid exchange between fractures and the rock becomes more significant.

Even with a method for accurately simulating the flow of fluids through the fracture network, researchers must still somehow convert field measurements into a set of discrete fracture network parameters such that the simulated flow model produces results that match observations from the field. This task of inverse modeling is known as history matching and researchers have deployed a wide range of methods to accomplish it. Methods include ensemble Kalman filters (Ping and Zhang, 2013; Nejadi et al., 2017), Markov Chain Monte Carlo optimization (Ma et al., 2008; Xu et al., 2013), and simulated annealing (Tran, 2007; Mahmoodpour and Masihi, 2016). An important method for converting a set of well observations into a hydraulic model for fractured rock is hydraulic tomography (Illman et al., 2009; Illman, 2014; Zha et al., 2015; Klepikova et al., 2020; Ringel et al., 2021). The process for hydraulic tomography begins by first acquiring data from pumping tests applied to a well field at the site of interest. These tests must be designed in a way that properly extracts information about the hydraulic properties of the rock between the wells. The tests usually involve selecting a well to pressurize, then monitoring the pressure of adjacent wells through time to see how the pressure from the injection well propagates through the rock. Once data is collected, a variety of inversion methods can be used to convert the collected data into hydraulic models. For Klepikova et al. (2020), a misfit function was first defined to measure the differences between the simulated and observed. The parameters of the model were then tuned using the Nelder–Mead Simplex algorithm (McKinnon, 1998) in order to find the optimal parameter values that minimizes the misfit function. Zha et al. (2015) performed their inversion step using the simultaneous successive linear estimator algorithm (Xiang et al., 2009) where a successive Bayesian linear estimator derives the mean parameter fields using the results of field tests and assumes prior knowledge of the mean value and spatial structures of estimated parameters. To perform inversion for a three-dimensional discrete fracture network, Ringel et al. (2021) used a reversible jump Markov Chain Monte Carlo method (Fan and Sisson, 2011) to calculate the likelihood a given change to a proposed model (such as adding or removing a fracture) yields a new model that is most likely to produce pressure transients that match with field tests. Note that many of these methods have the same underlying process of running multiple fluid flow simulations to find a group of model parameters that produce results that fit field observations. Often these simulations are large and computationally expensive, so researchers use a variety of techniques and heuristics to efficiently find solutions while minimizing the required amount of computational resources.

Another popular family of methods used by researchers includes evolutionary techniques, which are optimization algorithms with heuristics that are inspired by biological processes in nature. One of the most popular heuristics is the concept of natural selection. Possible solutions to an optimization problem are represented as individuals in a population. The fitness of these individuals is determined by how well they solve the optimization problem. Individuals that perform poorly are assigned a low fitness value and removed from the population. But individuals that perform well are assigned a high fitness value and are allowed to stay in the population. After the weakest individuals have been removed from the population, the remaining individuals then are allowed to reproduce and create a new generation of possible solutions. During the reproduction process, the child solutions adopt characteristics from each of the parent solutions. The idea is that the children will adopt the beneficial characteristics of their parents, ideally producing a new generation of solutions that will perform better than the previous generation. Evolutionary methods such as differential evolution (Das and Suganthan, 2010; Pant et al., 2020) and genetic algorithms (Mitchell, 1998; Sivanandam and Deepa, 2008) rely on this main heuristic of natural selection. Researchers then can use these techniques to find the optimal set of fracture network parameters that fits simulation results to field observations (Cadini et al., 2013; Liu and Reynolds, 2019; Zhang et al., 2019; Maucec et al., 2020).

Although it is important to be able to generate a set of possible fracture network parameters that produce simulation results that fit with field observations, it is just as important that the distribution of the fracture parameters also match what is observed in the rock. To effectively produce a set of fractures that imitate the parameter distributions in real life, stochastic fracture network generation methods can be used (J.P., 2005; Bonneau et al., 2013). These methods often deploy simple generating heuristics inspired by the rock mechanics involved with fracture formation and propagation. These simple rules can generate fractures with parameters such as fracture shape, size, orientation, or spatial density with the same distributions as observed in the field using techniques like well logging and micro-seismics (Willis et al., 2006; Han et al., 2021; Kennedy et al., 2022). Because fracture networks involve complexity at multiple scales, fractal dimensions are another popular method used to ensure that generated fracture models imitate distributions found in the field (Liu et al., 2015; Zhang et al., 2019). Overall, researchers have many methods available for them to produce realistic and accurate distributions of fractures.

Though seismic surveys can be used to generate a discrete fracture network (Sicking and Malin, 2019), this does not necessarily mean the generated model will produce simulated well test results that will match with all field observations such as well pressure tests. Instead, it is best to create discrete fracture network models using results from both seismic surveys and well pressure tests (Mayerhofer et al., 2006). For example, seismic survey data can used to define the number of fractures and the density of fractures for a given study volume. Hydraulic tomography can then use well pressure tests to determine remaining model parameters such fracture connectivity, orientations and size. Once discrete fracture networks can be generated to have some of the same parameter distributions measured from seismic surveys, it can be challenging to also modify the generate models such that they also match the flow characteristics measured from pumping and tracer tests performed in the field. When naively changing the parameters of the fracture network to better fit with measured flow characteristics, this often can cause the distribution of fracture parameters to deviate from distributions measured in the field. To deal with this need for a fracture model with both accurate fracture distributions and accurate flow characteristics, researchers use optimization techniques that can handle multiobjective functions (Maucec et al., 2020). Instead of using techniques that iteratively generate a population of fracture models that converge toward both accurate distributions and flow characteristics at the same time, this work proposes a different method that can meet this multiobjective goal in two distinct and straightforward steps. The first step involves establishing a method for readily generating a set of fracture models with fracture parameter distributions that match field observations. This step can be done by simply using any of the popular methods in the literature such as stochastic generation or using fractal dimensions. The second, critical step is to use the population of generated fracture networks to create new fracture models that are more likely to match flow characteristics but in a way that preserves the fracture distributions of the previously generated fracture network models. This is done by creating a genetic algorithm optimization architecture that is designed to preserve the fracture parameter distribution of the population with each generation. This work includes a synthetic experiment to investigate the effectiveness of this technique.

2. Methodology

2.1. Discrete fracture network

To model the discrete fracture networks, the software package DFNWorks by the Los Alamos National Laboratory was used (Hyman et al., 2015). DFNWorks is composed of multiple software packages combined to form a seamless software suite for generating three-dimensional

models of discrete fracture networks as well as simulating flow and transport of particles through these fracture networks. To generate a discrete fracture network, DFNWorks first uses a feature rejection algorithm for meshing (FRAM) method to generate the underlying geometric model for the discrete fracture network. At this stage, fractures are represented as flat polygons of various sizes and are distributed across a three-dimensional volume. Each of the polygons is assigned additional attributes such as aperture size. To generate a fracture network, the distribution of the fracture network parameters are first defined. Once defined, the distributions then are sampled and the fracture network is iteratively built, one fracture at a time. After each fracture is added, the FRAM method checks if a series of constraints are met. Constraints such as proximity to other fractures or proximity to the model domain boundary are considered. If the added fracture fails to meet a criterion, then the fracture is rejected and a new one is created. Otherwise, the fracture is kept within the fracture model and the process continues. When the growing fracture model achieves a certain stopping criterion, such as a set number of fractures, the iteration loop halts, and the geometric fracture model is presented as the final model ready for the next step of the process. After the geometric fracture model is created, the LaGriT meshing tool converts the geometric model into a mesh model. The meshing process produces a Delaunay triangulation mesh that is ideal for parallel computations.

During the fracture generation phase, the user has the freedom to define a probability distribution from which the software can sample to generate the fractures. For fracture radius, DFNWorks provides options for a log-normal distribution, an exponential distribution, a truncated power law distribution, or simply a constant. For this study, the truncated power law distribution was used:

$$P(r) = \frac{\alpha}{r_0} \frac{(\frac{r}{r_0})^{-1-\alpha}}{1-(\frac{r_u}{r_0})^{-\alpha}} \tag{1}$$

In this equation, the probability density function $P(r)$ is parameterized with the maximum fracture radius r_u, the minimum fracture radius r_0, and an exponent α that defines the shape of the power law distribution. The truncated power law distribution was used because its additional parameters allow greater control on the shape of the distribution that controls the fracture radius (Di Federico and Neuman, 1997; Di Federico et al., 1999; Neuman, 2008). But the proposed method does not rely on a specific distribution to function, so the approach will still work when an alternate fracture distribution function is used. For the fracture orientation, the orientation vector (normal to the fracture plane) must be sampled by essentially using a Gaussian distribution that is embedded onto the surface of a three-dimensional sphere. This was done by using the three-dimensional von Mises–Fisher distribution.

$$f(\boldsymbol{x};\boldsymbol{\mu},\kappa) = \frac{\kappa\ exp(\kappa \boldsymbol{\mu}^T \boldsymbol{x})}{4\pi \sinh(\kappa)} \tag{2}$$

In this equation, the probability distribution function $f(\boldsymbol{x})$ is parameterized by $\boldsymbol{\mu}$ and κ. $\boldsymbol{\mu}$ is the vector for the mean orientation, with T indicating that the vector is transposed. κ is the parameter that represents the variance of the distribution around the mean orientation. DFNWorks samples this distribution by using the algorithm outlined by Wood (1994).

2.2. Flow and transport simulation

The DFNWorks suite (Hyman et al., 2015) also includes software for running flow and transport simulations using the discrete fracture network. Called DFNFlow, the software takes the meshed fracture network model and uses PFLOTRAN to compute the pressure field across the network. PFLOTRAN (Lichtner et al., 2015) is an open source code base that was written by core developers from the U.S. Department of National Laboratories such as the Los Alamos National Laboratory, Sandia National Laboratory, Lawrence Berkeley National Laboratory, and Oak Ridge National Laboratory, as well as contributions from universities and research labs around the world. PFLOTRAN is capable of massively parallel operations that can operate at multiple scales and physics. PFLOTRAN also can solve differential equations for non-isothermal multiphase flow, reactive transport, and geomechanics in porous media. This study uses PFLOTRAN to solve for the single phase flow across the fractures during steady state. PFLOTRAN does this by solving the three-dimensional Richards equation (Richards, 1931). The mixed form of the Richards equation proposed by Celia et al. (1990) yields robust numerical solutions and maintains mass balance for unsaturated flow problems (Wu et al., 2021):

$$\frac{\partial}{\partial t}(\phi s \eta) + \nabla \cdot (\eta \mathbf{q}) = Q_w \tag{3}$$

In this equation, t is time, ϕ is the porosity of the soil matrix, s is the water saturation, η is the molar water density, $\mathbf{q}$ is the Darcy flux, and Q_w is a (positive) source or a (negative) sink of water. The Darcy flux, $\mathbf{q}$, is calculated using Darcy's Law:

$$\mathbf{q} = -\frac{k k_r}{\mu} \nabla (P - W_w \eta g z) \tag{4}$$

where k is the intrinsic permeability, k_r is the relative permeability, μ is the water viscosity, P is the pressure, W_w is the formula weight of water, g is the acceleration of gravity, and z is the vertical component of the position vector. In this study, the van Genuchten soil water retention curve (Van Genuchten, 1980) and the Mualem relative permeability function (Mualem, 1976) was used to calculate the water saturation and the relative permeability. For this study, the boundary conditions for the model will be a no-flow boundary condition for all boundaries except for the injection and extraction well. The two wells will be set to a boundary condition of a constant pressure. The permeability of the fractures are derived as a function of the aperture of the fracture. This is done by first assuming the fluid flows between two smooth, impermeable, parallel plates. This assumption allows the Boussinesq equation (Boussinesq, 1868) to yield the volumetric flow rate Q per unit fracture width normal to the direction flow (Hyman et al., 2016):

$$Q = -b^3 \frac{\rho g}{12\nu} \nabla h \tag{5}$$

In this equation, b is the fracture aperture, ρ is the fluid density, g is gravitational acceleration, and h is hydraulic head. This relationship between fracture aperture and flow rate can be used to derive a relationship between aperture and transmissivity (Hyman et al., 2016):

$$T = b^3 \frac{\rho g}{12\nu} \tag{6}$$

This equation is referred to as the cubic law (Witherspoon et al., 1980). PFLOTRAN uses this relationship to convert fracture aperture into permeability. Note that these equations assume that the flow through the fracture is homogeneous and that there is no fluid exchange between the fracture and the matrix. Such an assumption can be justified when modeling flow through impermeable crystalline rock. For this study, focus is on investigating whether the genetic algorithm is capable of tuning a model with its flow dynamics dominated by the geometry and connectivity of the fractures. However, the methods developed in this work can be extended to study sites where there is significant fluid exchange between the fracture and the matrix. After the flow and pressure field is calculated, this information is sent to the software called DFNTrans for particle transport calculations. During this step, DFNTrans adopts the Lagrangian approach for calculating the path that a cluster of non-reactive, indivisible particles take through the discrete fracture model. To calculate the path the particles take, the fluid velocity field first is calculated using the pressure field result from PFLOTRAN as well as other flow attributes derived from the fracture network. To calculate the velocity field, PFLOTRAN was set to use a constant porosity of 25%. Since the goal of this work was to test whether the proposed method can tune a model with flows dominated by fracture geometry and connectivity, porosity was simply

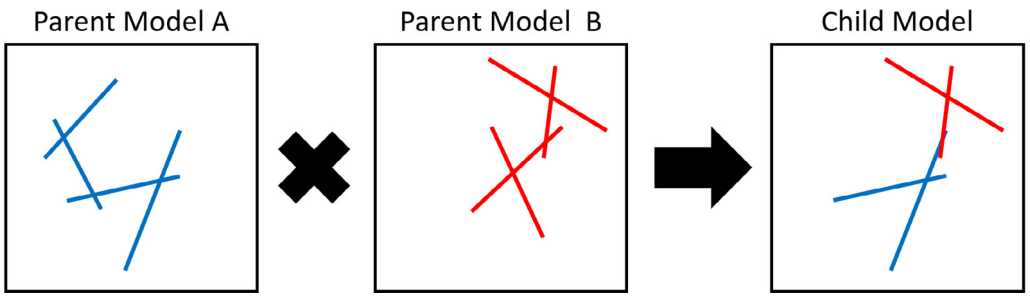

Fig. 1. An example of genetic mixing for the genetic algorithm adapted to discrete fracture networks. Each of the fractures of a model is represented as individual genetic bases that compose the genome of the model. During genetic mixing, the genome of the child model is created by randomly copying the fractures (the genetic bases) from each of the parent models. Doing this randomly allows fracture parameter distributions such as fracture orientation to be preserved during the creation of the child model. To ensure the population performance improves with each generation, the parent models are generally high-fitness members of the population.

set to a constant. The proposed method also does not rely on a constant porosity, so it can be extended to handle distributions where porosity does become a function of aperture. After the velocity field is solved, the particle paths can be calculated by numerically integrating the trajectory equation for each particle. At the intersection of fractures, it is assumed that complete mixing occurs. This means when a traveling particle meets an intersection, a flux-weighted stochastic method is used to determine whether or not the particle stays within the fracture plane or would flow into the intersecting fracture plane. Because of the stochastic nature of flow through fracture intersections, the general flow structure of the entire fracture network can be studied by using many tracer particles or with multiple runs of the transport simulation.

2.3. Genetic algorithm for fracture networks

Genetic algorithms refer to a class of optimization algorithms in which a family of optimal solutions is found by iteratively applying the process of natural selection. In this work, we show that genetic algorithms can be used to tune a DFNM where the genetic algorithm is tasked with finding the correct distribution of fractures such that it produces simulated tracer test results that match with observations. The process begins with generating an initial population of potential solutions to an optimization problem. Each member of the population is ranked with a fitness function. A fitness function is a function designed to convert the performance of single candidate solution into a single number. This fitness function is used to rank members of the population (candidate solutions) by their ability to perform the task. Solutions that do a superior job of optimizing the given problem will receive a high fitness score, while solutions that do a poor job solving the optimization problem are given a low fitness score. Below a certain fitness threshold, solutions with a lower fitness score are removed from the population. The remaining solutions then are used to generate a new generation of potential solutions. This is done by selecting high-fitness parents and mixing their genetic information to produce new children solutions with a greater chance of yielding a high-performance score. After repeating this process for multiple generations, the population evolves into a family of high-fitness solutions that effectively solve the optimization problem.

One important factor in a successful genetic algorithm is having a well-designed method for encoding the solutions as a genetic code. If done correctly, the genetic mixing process can properly explore the solution space and quickly find the optimal solution. Within the context of discrete fracture networks, this study proposes to represent each fracture as a single genetic base within the complete genetic code of a discrete fracture network model (Fig. 1). For example, a discrete fracture network with 300 fractures will have a genetic code that is 300 bases long. To create new discrete fracture networks, the genetic code of the child discrete fracture network will be composed of the genetic bases (fractures) randomly sampled from the genetic code of high-fitness parent discrete fracture networks. The key idea is that this method of genetic mixing preserves the distributions of the fracture parameters. That means if the genetic algorithm process begins with a fracture network model population that was all generated with the same parameter distribution, then after multiple generations of applying the genetic algorithm, the final population will have the same fracture parameter distribution. The main difference between the initial and final generation is that the final population of discrete fracture networks will be able to produce simulated flow behaviors that best match with field observations.

Note that the genetic mixing process does not create new fractures. After many iterations of the genetic algorithm, the population becomes filled with copies of the same genetic code. Although the genetic algorithm process will converge quickly, it will also have a high risk of converging prematurely and stopping at a suboptimal solution. To prevent the genetic algorithm from halting at a locally optimum solution, new fractures must be added to the population. The new fractures must be added in a way that does not change the parameter distribution of the fractures. This is done by adding newly generated fracture networks midway through the process. These new fracture networks are generated using the same parameter distributions as were used to generate the initial population. If these new fracture network models were simply added to the population, then these models would be removed quickly from the population because of a low fitness value that could not compete with models that have already evolved. To force the genetic influence of the newly added models, the genetic mixing process allows the combination of genetic code between high fitness models and newly generated models. By controlling the size and frequency the freshly generated models that are added to the population, the user can increase the likelihood that the genetic algorithm would converge to the global optimum.

A complete description of the discrete fracture genetic algorithm used in this study begins with generating a population of discrete fracture network models. For this study, a population of 20 models was maintained. Each of the fracture network models was generated with the same parameter distribution for the fracture radius and fracture orientation. A fitness value then was calculated for each member of the population. For this study, fitness was determined by how well a given fracture network could recreate the time-versus-concentration curves of a tracer transport test measured on a reference fracture model. The individuals of the population then were ranked by their fitness value and a certain fraction of the lowest performing individuals was removed from the population. To maintain the size of the population, newly generated discrete fracture models then were added to the population. Some of these new models were generated using the same parameter distributions as the initial generation. Other new models were generated by randomly selecting two parent models, then randomly copying the genetic code from each of the parent models to produce the genetic code for the child model. The length of the genetic code of the child model (the number of fractures in the child model) was set randomly as a number between the number of fractures for each of the parent

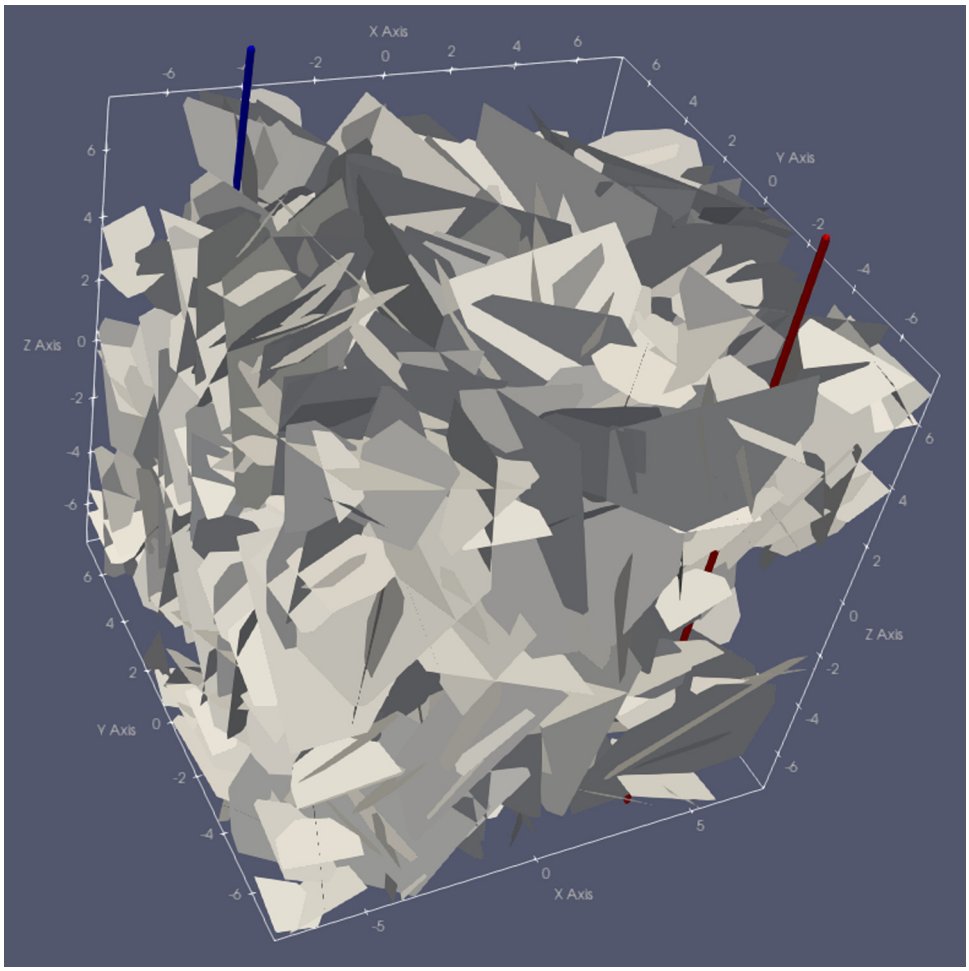

Fig. 2. An example of a generated discrete fracture model. For each generated model, 2100 discrete fractures were randomly placed within a cubic model domain with a side length of 15 m. The fracture orientations were determined by sampling the three-dimensional von Mises–Fisher distribution. Two wells were vertically inserted along the diagonal of the model. Water was injected into the model through the red well and was extracted from the model through the blue well. The number of fractures was selected to ensure that a randomly generated fracture network would be likely to have the two wells hydraulically connected through the discrete fracture network.

models. For example, for parent models with 180 and 200 fractures, then the generated child model could have anywhere from 180 to 200 fractures. Newly generated models were added until the number of models was equal to the set population number. As an example, for a current generation of 20 models, the next generation could contain ten copies of the best models from the previous generation, six new models generated using genetic mixing, and four models generated from sampling the same parameter distribution as used to generate the initial population. The fitness of the new generation then was calculated and the entire process can be repeated multiple times to create many generations. The entire loop then could be halted when the fitness of subsequent generations no longer improved. A complete summary of the genetic algorithm is presented in Algorithm 1. A flowchart of the process is also presented on Fig. 4.

2.4. Synthetic fracture model experiment

For this study, the discrete fracture genetic algorithm was tested on a synthetic fracture model in which tracer particles were injected, simulated to flow through the fracture network, and extracted through two wells (Fig. 2). The domain of the fracture model was in the form of a cube with a side length of 15 m. All the sides of the domain were set to a no-flow boundary condition. There was also no water exchange between the fractures and the matrix. Water only flows through the fractures. Water was only allowed to enter and exit the model through two wells that intersect the model domain. The two wells were vertically oriented, centered along the diagonal of the model, and spaced 14 meters apart. To create flow, the wells were set with a constant pressure difference of 4.0×10^6 Pascals. To generate the discrete fracture network within the model domain, the DFNGen function (which uses FRAM and LaGriT) of DFNWorks was used. In the geometric model, fractures were modeled as two-dimensional octagons with their radius and orientation determined by sampling defined parameter distributions. The fracture radius was sampled from a truncated power law distribution using a maximum fracture radius of $r_u = 5.0$ meters, a minimum fracture radius of $r_0 = 1.0$ meters, and an exponent of $\alpha = 2.6$ that defines the shape of the power law distribution. The fracture orientation was sampled using the three-dimensional von Mises–Fisher distribution with the mean orientation vector $\boldsymbol{\mu}$ set to a vertically oriented normal vector, and the orientation variance set to $\kappa = 1.0$. The aperture of the fractures was set to a constant width of 1.0×10^{-5} meters. Note that a constant fracture aperture is a valid simplification for this study. This work aims to test how a genetic algorithm can handle discrete fracture network models where the flow is mainly governed by how the fractures are connected to each other. To focus on this mechanism, all fractures were given a constant width. The proposed method can handle variations in fracture aperture, but this simplification is valid for initial tests of the proposed method. DFNGen was instructed to randomly insert 2100 fractures into the model domain. This number of fractures was determined by the computational time and resources available for this work. For the given model dimension and constraints, this was the minimum number of

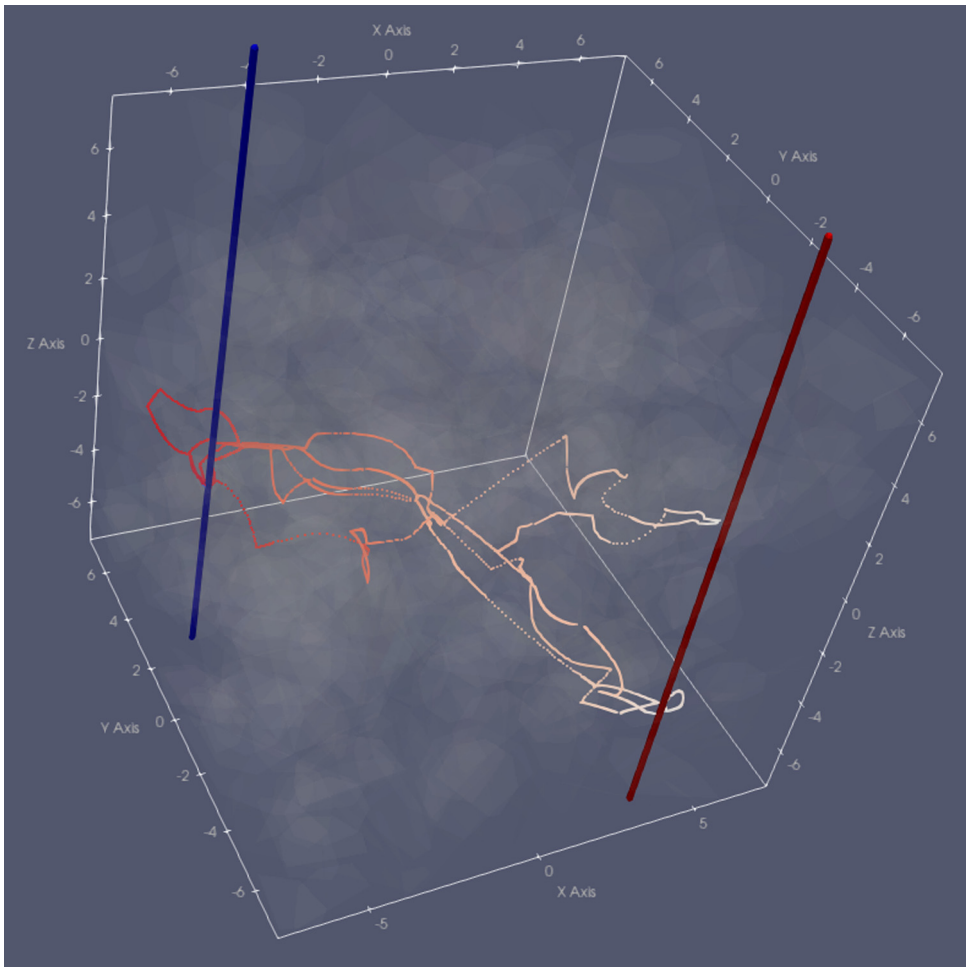

Fig. 3. An example of a transport simulation running on a generated discrete fracture model. Tracer particles were first injected into the model through the red injection well. Ten particles were placed at every point a fracture intersected the injection well. After the tracer particles moved through the fracture network, they were collected from the blue extraction well. Shown are some of the paths the particles took through the fracture network, with their color indicating the time spent within the network. At each fracture intersection, the next path the particles took was a stochastic process weighted by the local flux. After tracer particles arrived at the blue extraction well, their total travel time was recorded and then compiled to create the breakthrough curve for a given model.

fractures needed to study how the genetic algorithm would perform when tuning a discrete fracture network model. The proposed technique can be scaled to handle models with a larger number of fractures. This number of fractures was also chosen to ensure that any randomly generated fracture model would be likely to hydraulically connect the two wells. To run the particle transport simulation, the flow simulation was run first until it reached steady state. Afterward, tracer particles were added to the model through the injection well (Fig. 3). Ten tracer particles were added to every point where a fracture intersected the wells. DFNTrans then calculated the trajectory of the particles. As the transport simulation progressed, DFNTrans recorded the time it took for each particle to reach the extraction well. The transport simulation ended when all particles reached the extracting well. The recorded arrival times then could be used to create the breakthrough curves for the given model. Note that the breakthrough curve derived using arrival times of simulated particles can be used as a proxy for the cumulative molar amount of tracer recovered at the extraction well recorded over time. For example, recovering 25 out of 50 simulated tracer particles collected over the span of one simulated week would be the same as recovering 0.5 moles out 1.0 moles of an injected tracer compound collected over a span of a week.

To select the reference model, one of the randomly generated fracture models was chosen as the reference model. The goal of the genetic algorithm is to look for a discrete fracture model that produces the same breakthrough curves as the reference model. To do this, the genetic algorithm needs a fitness function that can rank how well each of the breakthrough curves matches with the breakthrough curves of the reference model. This is done by first calculating the quantiles of the arrival times of the tracer particles. For this study, 11 quantiles were calculated: 0%, 10%, 20%, and so on up to 100%. This converts the breakthrough curve into an 11-dimensional vector. To calculate the fitness function, the L2 distance is calculated between the quantile vector of the tested model and the reference model. This means taking the difference between the quantile vectors for the two models, summing the squares of all the elements of the new difference vector, and finally taking the square root of this final sum. Note that this metric is an error value. Models that yield a low value are the best fit with the reference model and so are most likely to be kept in the population. Conversely, models with a high value poorly match the reference model, so they are most likely to be removed from the population. After the fitness of each model is evaluated, half of the population's worst-performing models are removed from the population. Newly generated models then are added until the number of models in the population is the same as the starting population. For this study, the population was initialized with 20 models. With each new generation, ten models are copies of the previous generation's best models, nine are newly generated from genetic mixing of models from the previous generation, and one model is generated using the same parameter distribution as was used to create the initial population. During model generation, heavy emphasis was placed on genetic mixing because this ensures the

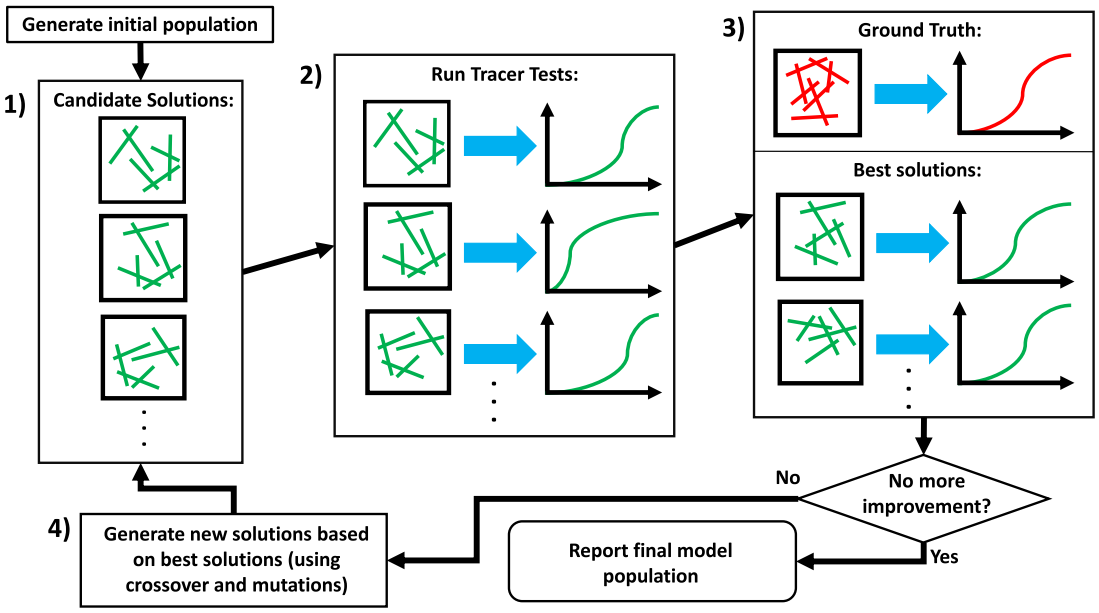

Fig. 4. A flowchart for the work flow of applying genetic algorithms to calibrating discrete fracture networks. The genetic algorithm begins by first generating a population of possible solutions. These solutions are a series of randomly generated discrete fracture network models where each model has a different placement, geometry and number of fractures (Fig. 4.1). After preparing the population of models, each model is evaluated using a simulated tracer test (Fig. 4.2). Using PFLOTRAN, tracer particles are added through the injection well and removed from the extraction well. By recording the time it takes for each particle to move from the injection well to the extraction well, a breakthrough curve can be generate for each model. After breakthrough curves have been calculated for all models, all models with breakthrough curves that are too different from the breakthrough curve calculated using the ground truth model are removed from the population (Fig. 4.3). The remaining models in the population have curves that best match with the simulated observed data from the ground truth model. Note that the ground truth model is a defined discrete fracture model that remains constant throughout the experiment. Running the simulated particle tracer test on the ground model yields a breakthrough curve declared to be the simulated observation data. After keeping the models with curves that best match the simulated observed data, new models are generated using the current population of the best models (Fig. 4.4). These models are generated using genetic algorithm techniques such as crossover (Fig. 1) and mutations applied to the fractures of the network. Once a new generation of models is prepared, the entire workflow repeats until there is no improvement in the fitting quality of the models. At the end of the loop, this process yields a population of discrete fracture network models that produce breakthrough curves that best match the simulated observed breakthrough curves of the ground-truth model.

genetic algorithm can converge quickly toward optimal solutions. The genetic evolution process was repeated until no further improvement was observed from subsequent generations. In this study, the process continued for 40 generations. For this study, it took 11 days to complete the entire experiment, with DFNWorks running single-threaded on an AMD Ryzen 3900X processor. The runtime for this method can be significantly reduced by running multiple parallel processes when calculating the breakthrough curves for each of the candidate solutions (Fig. 4.2). Since genetic algorithms are a method that can easily take advantage of parallel processing, this method can computational scale as well as other methods.

3. Results

During the genetic algorithm loop, the performance of the models within each generation was recorded. Fig. 5 plots the distribution of the model performances within each generation of 20 models. The graph includes the 10%, 50%, and 90% quantiles of the distribution. The model error value is the value from the fitness functions. Recall that this fitness function is based on the differences in the tracer arrival time quantiles between the tested reference model. The graph shows that starting with a median model error of 3.5, successive iterations of the genetic algorithm led to a population of models with a median model error of 0.75. The population reached this value by 15 generations. Beyond 15 generations, the population performance did not improve but instead remained at this performance level. Fig. 6 shows that the variance of a population's model performances also evolved throughout the iterations of genetic evolution. At the start, the initial generated models had a model error variance of 0.4. Then during the genetic algorithm process, each iteration yielded a very different variance value. Although the variance of the model error fluctuated widely from generation to generation, overall the population's variance trended downward. The downward trend stopped at generation 15, the same generation that the median model error reached its steady state value. After 15 generations, the variance no longer decreased but instead stayed at a value of 0.09. The volatility of the model error variance also decreased substantially beyond 15 generations.

During the experiment, the breakthrough curves for each of the models for each generation were recorded. Fig. 7 shows the breakthrough curves recorded for the first generation, the final generation, and the reference model. The breakthrough curves are jagged in appearance because of the relatively small number of tracer particles used. Such simulated breakthrough curves are expected to be more smooth with the use of more tracer particles. Note that for the initial population of generated models, the majority of the models had an early breakthrough curve, with the median of the curves having their 50% point at 2.4×10^{-3} years. To test the genetic algorithm's ability to generate models with behaviors outside of the initial generated distribution, note that the selected reference model has a breakthrough curve with a 50% point at 8.4×10^{-3} years. After 40 generations of applying the genetic algorithm, the final population of models successfully shifted right, toward the breakthrough curve of the reference model. After 40 generations, the median of the curves had their 50% point occur at 8.5×10^{-3} years, which is essentially the same as the breakthrough curve for the reference model. Although the final curves fit well at the 50% point, the slope of the final breakthrough curves did not match well with the curves of the reference model. At the particle recovery amount of 10%, the reference model's breakthrough curve reached this point at 2.8×10^{-3} years, yet after 40 generations the median of breakthrough curves reached this point at 4.0×10^{-3} years. For the tracer recovery percentage of 90%, the reference model's breakthrough curve reached this percentage at 3.7×10^{-2} years, yet after 40 generations the median of breakthrough curves reached this percentage at 2.4×10^{-2} years.

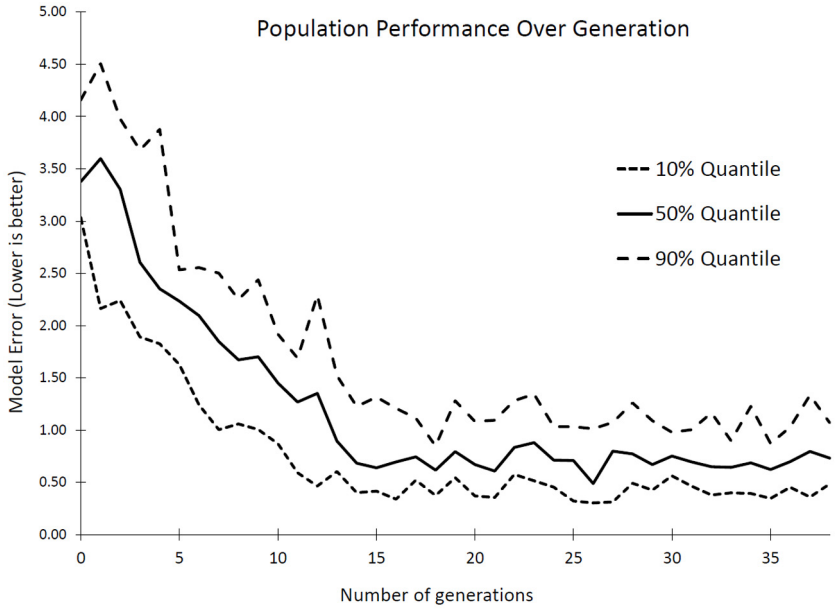

Fig. 5. A plot of the general performance of the model population over multiple generations. The *y*-axis is the overall model error, a metric based on how close a given model could generate a breakthrough curve that matched the breakthrough curve of the reference model. Plotted are three quantiles of the population's performance: 10%, 50%, and 90%. These curves show that as the discrete fracture genetic algorithm progressed, the overall error of the population of models reduced until it reached a limit of 0.75 after 15 generations.

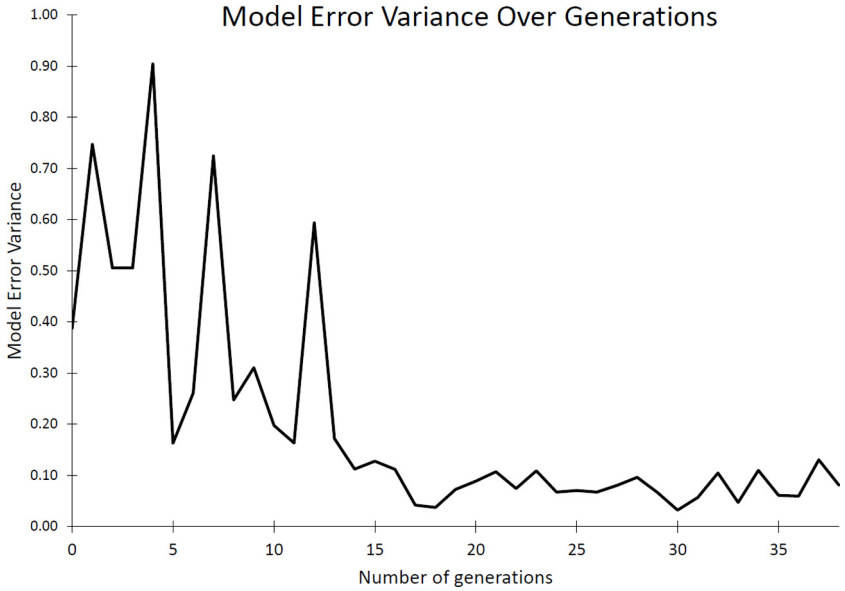

Fig. 6. A plot of the model error variance of the population through the generations. The plot shows that through successive generations, the genetic algorithm reduced the variance of the population until it reached a minimum value of 0.09 after 15 generations. Note that before generation 15, the model error variance moved erratically as the general trend progressed downward. After generation 15, the model error variance fluctuated slightly around the plateau value.

4. Discussion

Overall the results show that the genetic algorithm was able to successfully evolve the population to produce discrete fracture models with tracer breakthrough curves that matched with an observed breakthrough curve. Fig. 5 shows that the algorithm was able to reach convergence to a set of optimal solutions within 15 generations by modifying a population of 20 models. Note that the population model error decreased until it reaches a limit of 0.75. Recall that with hundreds of fractures, and with each fracture having its own set of parameters, the overall discrete fracture model was heavily parameterized. This means that the genetic algorithm theoretically should have the ability to produce discrete fracture models with breakthrough curves that perfectly match the curve of the reference model and achieve a model error of zero, yet the algorithm only achieved a minimum of 0.75. One reason why the genetic algorithm failed to achieve a lower model error is that the simulation of particle transport was not completely deterministic. If the particle transport simulation were run twice on the same discrete fracture model, it would yield slightly different results. The reason is that, at every fracture intersection, the path the particle would take is a stochastic process with its likelihoods weighted by flux. Because the fitness function was calculated by using the results of the particle transport simulation, the fitness function adopted its stochastic value. So evaluating the fitness function on the same model multiple times produced different results. The uncertainty of the function limited the genetic algorithm's ability to find the most optimal model, so the

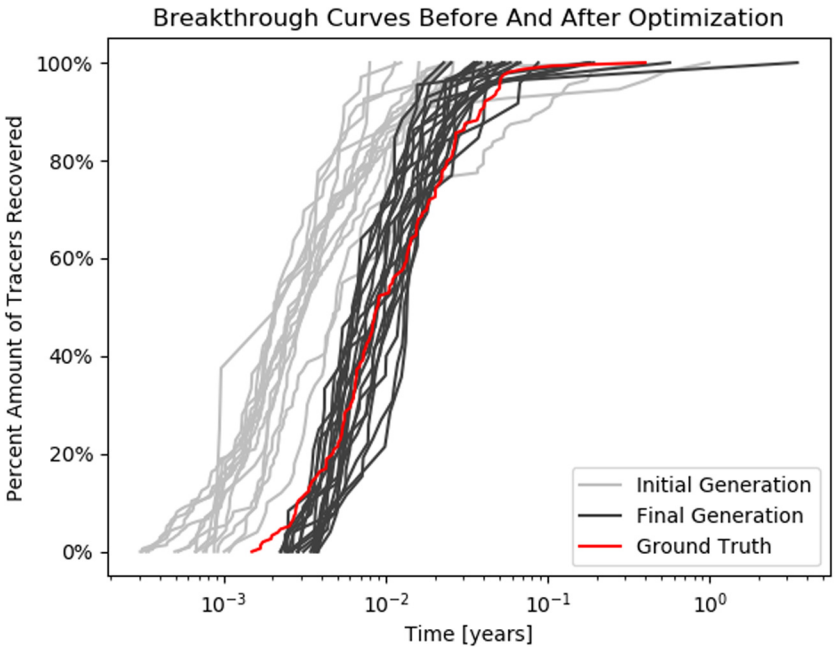

Fig. 7. A plot of the breakthrough curves for the initial population, the final population, and the reference model (ground truth). The breakthrough curves were calculated by recording the travel times for each of the tracer particles, then presenting the data as the percentage of total particles recovered, plotted over time. The curves were plotted on a logarithmic time scale. The red line shows the breakthrough curve of the reference model with a 50% recovery time of 8.4×10^{-3} years. The light gray lines show the initial population of breakthrough curves with a median 50% recovery time of 2.4×10^{-3} years. The dark gray lines show the breakthrough curves of the populations after 40 generations of the genetic algorithm, with a 50% recovery time of 8.5×10^{-3} years. Note that the genetic algorithm was able to successfully shift the breakthrough curves of the population to match the overall arrival times of the curves of the reference model, but it struggled to match the slope of the breakthrough curve of the reference model.

genetic algorithm generated a population of models that had a high likelihood of producing a good fitness score. The uncertainty associated with the transport simulation can be reduced by either increasing the number of particles added to the simulation or by running the transport simulation multiple times and recording the average result. Another reason why model error did not reach zero is because of forcing the genetic mixing between high fitness models and models that are newly generated. Recall that to prevent the genetic algorithm from prematurely stopping at a local optimum, genetic diversity was introduced to the population by forcing the high-fitness models to genetically mix with models that were newly generated when producing new child models. As a population of optimal models evolves, the genetic variation of the models gets reduced. And at every step, genetic diversity is injected into the population from newly generated discrete fracture models. At a certain point, the reduction in genetic diversity caused by removing the least fit models is equal to the genetic diversity added by the newly generated models, and so the genetic diversity reaches a steady state. The aforementioned reasons also explain why the variance of the model error did not reach zero (Fig. 6). The stochastic nature of the fitness function and the consistent addition of genetic diversity to the population prevented the variance of the model error from reaching zero.

The breakthrough curves of Fig. 7 also show that the genetic algorithm was able to adjust the models successfully so that they produced curves that better followed what was produced by the reference model. After 40 generations, the breakthrough curves shifted toward the right, meaning that the tracer particles arrived at the extraction well later than earlier generations of models. The genetic algorithm achieved this by adjusting populations of fracture parameters until the bulk hydraulic conductivity of the fracture network was increased. This led to a lower overall flow rate and so a later arrival time for the particles. Fig. 5 also shows that the average slope of the breakthrough curves of the final generation did not match the slope of the breakthrough curve of the reference model. For the first 10% of the arriving particles, the particles in the final generation models arrived later than the particles for the reference models. For the last 10% percent of the arriving particles, the particles in the final generation models arrived earlier than the particles for the reference model. This behavior can be attributed to the uncertainty of the fitness function. During the start of the genetic algorithm, the fitness functions could produce an error signal that was much greater than the uncertainty of the fitness function. This allowed the genetic algorithm to quickly distinguish which members of the population were high-performing. At this stage, errors such as the temporal shift of the breakthrough curve could be fitted quickly. But as evolution progressed, the fitness function produced smaller error signals until, finally, the error signal was smaller than the noise generated by the fitness function. At that stage, the slopes between breakthrough curves became difficult to distinguish from each other, thereby inhibiting further improvement.

One limitation for the results of these experiments is that there is no guarantee that the calibrated models will generalize beyond the specific placement of the test wells during the calibration process. This means that if a tracer test was performed on the calibrated model with the test wells in a new orientation, then the resulting breakthrough curve may be different than the breakthrough curves from applying the tracer test on the ground truth model with the same new orientation of the test wells. Since information about the hydraulic structure of the fractures comes solely from hydraulic tests with the wells, the tracer tests essentially become blind to any regions in the fracture network that are not hydraulically connected to the wells. A similar limitation applies to fracture networks which are anisotropic. If tracer tests were performed on wells that were installed in an anisotropic fracture matrix in only a single orientation, rotating the wells by 90 degrees would yield a different breakthrough curve. To reduce the risk of discrete fracture models overfitting to the biases introduced by well placement, multiple wells can be installed in multiple orientations. This allows for a more comprehensive measurement of the hydraulic structure of the fracture network and helps to mitigate the limitations of wells installed in a single orientation.

Algorithm 1: Genetic Algorithm for Discrete Fracture Networks

Set: N_{total} = The total number of models in the population
Set: $N_{children}$ = Number of models generated by genetic crossover
Set: $N_{generated}$ =
Number of models generated by sampling a parameter distribution Ensure: $N_{total} > N_{children} > N_{generated}$
Define: $F(m)$ = Error of a given model m. To be minimized
Define: $N_{fractures}(m)$ = Number of fractures in model m
Define: $round(r)$ = Round number r to the nearest integer
begin
  Generate a population of N_{total} models
  while *Stopping Criteria Not Met* **do**
    Evaluate $F(m)$ for all models in population
    Sort population of models m by their error $F(m)$
    Remove $N_{children} + N_{generated}$ models from the population with the highest error
    for $i = 1, 2, \cdots, N_{generated}$ **do**
      Generate a model m by sampling a parameter distribution
      Add model m to the population
    for $i = 1, 2, \cdots, N_{children}$ **do**
      Randomly select models m_1, m_2 from the population such that $m_1 \neq m_2$
      Prepare empty model m_3
      Let: u = Random rational number between 0 and 1
      for $i = 1, 2, \cdots, N_{fractures}(m_1) \times round(u)$ **do**
        Select a random fracture f in m_1 such that f does not exist in m_3
        Add a copy of fracture f to m_3
      for $i = 1, 2, \cdots, N_{fractures}(m_2) \times (round(1-u))$ **do**
        Select a random fracture f in m_2 such that f does not exist in m_3
        Add a copy of fracture f to m_3
      Add model m_3 to the population
  Return population of models as final result
end

5. Conclusion

This study introduces a method for generating a population of discrete fracture models that produce simulated results that match with field observations. The method can achieve this model-tuning capability without changing the distribution of the fracture parameters. This allows the method to not only produce models with flow characteristics that match field pumping tests but to do so in a way that creates a population of fractures that can match fracture distribution parameters observed by field seismic surveys. Note that this study does not use any data from a seismic survey. The key idea is that if there is prior knowledge about the distribution of a fracture parameter, then the proposed method can generate possible solutions that preserve this distribution. This prior knowledge of parameter distributions may come from analysis of seismic surveys but can also come from other methods. The method used in this study is the genetic algorithm modified in a way that can handle discrete fracture networks. This was done by encoding every model's fracture as a genetic base in the model's genome. During genetic mixing, the child model is generated by randomly copying genetic bases from each of the parent models. This process allows the fracture parameter distribution of the child model to be the same as the distribution of the parent models. To test how well a genetic algorithm modified for discrete fracture networks can perform, the genetic algorithm was applied to a synthetic case in which the goal was to find a population of discrete fracture models that, when run with particle transport simulations, can produce breakthrough curves that match the observed breakthrough curve from a reference model.

The results show that the genetic algorithm was able to successfully produce a population of discrete fracture models with breakthrough curves that closely match the reference model's breakthrough curve. Within the span of 15 generations, the genetic algorithm reduced the model error and variance to a minimum that was bounded by the uncertainty of the fitness function and by the algorithm injecting genetic diversity into the population. The genetic algorithm was found to excel at adjusting the fracture network's bulk hydraulic conductivity to temporally shift the breakthrough curve until it closely matched the breakthrough curve of the reference model. The genetic algorithm changed the bulk hydraulic conductivity of the model by adjusting the number of fractures, the orientation of the fractures, the orientation of the fractures and the location of the fractures. By changing these parameters, the connectivity of the fractures changes, thereby changing the bulk hydraulic conductivity of the model. The genetic algorithm also was found to struggle with adjusting the population's breakthrough curves to match the slope of the reference breakthrough curve. Many issues caused by the uncertainty of the fitness function can be resolved by increasing the number of particles used in the transport simulation or by re-running the simulation and using the average simulation result.

Future work for this study includes testing the discrete fracture genetic algorithm on models with a different fracture parameter distribution. For example, this study involved fractures that have a relatively even distribution of orientations. But there are subsurface reservoirs with fractures that are heavily biased toward one or two orientations. Such fracture models with multiple families of fracture orientations might change the effectiveness of the genetic algorithm. Another path of study is the development and testing of new fitness functions that are based on different pumping or tracer tests. Because this study found that the performance can be bounded by the uncertainty associated with the fitness function, future work could focus on developing better fitness functions for applications with discrete fracture networks. Related work also could involve development of better genetic mixing strategies. The current strategy randomly selects fractures to be copied over to the child model. This method ignores all the other fractures connected to the copied fracture, and so that connectivity information can be destroyed during the genetic mixing process. Future work could help develop a better genetic mixing scheme that considers the hydraulic connections made by adjacent fractures. Any future improvement of the discrete fracture genetic algorithm could help researchers quickly and more efficiently solve the important problem of inverse modeling of fractured subsurface reservoirs.

CRediT authorship contribution statement

Fleford Redoloza: Conceptulization, Methodology, Investigation, Writing – original draft. **Liangping Li:** Conceptulization, Supervision, Funding acquisition. **Arden Davis:** Writing – review & editing.

Declaration of competing interest

Liangping Li, Fleford Redoloza and Arden Davis declares that they have no conflict of interest.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work has been supported through a grant from the National Science Foundation (OIA-1833069). The authors wish to thank the associate editor as well as two anonymous reviewers for their comments, which substantially helped to improve the final version of the manuscript.

References

Bonneau, F., Henrion, V., Caumon, G., Renard, P., Sausse, J., 2013. A methodology for pseudo-genetic stochastic modeling of discrete fracture networks. Comput. Geosci. 56, 12–22.

Boussinesq, J., 1868. Mémoire sur l influence des frottements dans les mouvements réguliers des fluids. J. Math. Pures Appl. 13, 377–424.

Cadini, F., De Sanctis, J., Bertoli, I., Zio, E., 2013. Upscaling of a dual-permeability Monte Carlo simulation model for contaminant transport in fractured networks by genetic algorithm parameter identification. Stoch. Environ. Res. Risk Assess. 27, 505–516.

Celia, M.A., Bouloutas, E.T., Zarba, R.L., 1990. A general mass-conservative numerical solution for the unsaturated flow equation. Water Resour. Res. 26, 1483–1496.

Das, S., Suganthan, P.N., 2010. Differential evolution: A survey of the state-of-the-art. IEEE Trans. Evol. Comput. 15, 4–31.

Di Federico, V., Neuman, S.P., 1997. Scaling of random fields by means of truncated power variograms and associated spectra. Water Resour. Res. 33, 1075–1085.

Di Federico, V., Neuman, S.P., Tartakovsky, D.M., 1999. Anisotropy, lacunarity, and upscaled conductivity and its autocovariance in multiscale random fields with truncated power variograms. Water Resour. Res. 35, 2891–2908.

Fan, Y., Sisson, S.A., 2011. Reversible jump mcmc. In: Handbook of Markov Chain Monte Carlo. pp. 67–92.

Han, W., Wang, Y., Li, Y., Ni, X., Wu, X., Wu, P., Zhao, S., 2021. Recognizing fracture distribution within the coalbed methane reservoir and its implication for hydraulic fracturing: A method combining field observation, well logging, and micro-seismic detection. J. Nat. Gas Sci. Eng. 92, 103986.

Hyman, J., Aldrich, G., Viswanathan, H., Makedonska, N., Karra, S., 2016. Fracture size and transmissivity correlations: Implications for transport simulations in sparse three-dimensional discrete fracture networks following a truncated power law distribution of fracture size. Water Resour. Res. 52, 6472–6489.

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. Comput. Geosci. 84, 10–19.

Illman, W.A., 2014. Hydraulic tomography offers improved imaging of heterogeneity in fractured rocks. Groundwater 52, 659–684.

Illman, W.A., Liu, X., Takeuchi, S., Yeh, T.C.J., Ando, K., Saegusa, H., 2009. Hydraulic tomography in fractured granite: Mizunami underground research site, Japan. Water Resour. Res. 45.

J.P., Chilès, 2005. Stochastic modeling of natural fractured media: a review. Geostat. Banff 2004, 285–294.

Kennedy, H., Löer, K., Gilligan, A., 2022. Constraints on fracture distribution in the Los Humeros geothermal field from beamforming of ambient seismic noise. In: EGUsphere. pp. 1–30.

Klepikova, M., Brixel, B., Jalali, M., 2020. Transient hydraulic tomography approach to characterize main flowpaths and their connectivity in fractured media. Adv. Water Resour. 136, 103500.

Lichtner, P.C., Hammond, G.E., Lu, C., Karra, S., Bisht, G., Andre, B., Mills, R., Kumar, J., 2015. PFLOTRAN User Manual: A Massively Parallel Reactive Flow and Transport Model for Describing Surface and Subsurface Processes. Technical Report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States); Sandia ....

Liu, R., Jiang, Y., Li, B., Wang, X., 2015. A fractal model for characterizing fluid flow in fractured rock masses based on randomly distributed rock fracture networks. Comput. Geotech. 65, 45–55.

Liu, Z., Reynolds, A.C., 2019. History matching an unconventional reservoir with a complex fracture network. In: SPE Reservoir Simulation Conference. OnePetro.

Ma, X., Al-Harbi, M., Datta-Gupta, A., Efendiev, Y., 2008. An efficient two-stage sampling method for uncertainty quantification in history matching geological models. SPE J. 13, 77–87.

Mahmoodpour, S., Masihi, M., 2016. An improved simulated annealing algorithm in fracture network modeling. J. Nat. Gas Sci. Eng. 33, 538–550.

Maucec, M., Zhang, S., Meza Camargo, O., Olukoko, O., 2020. New approach to history matching of simulation models with discrete fracture networks. In: International Petroleum Technology Conference. OnePetro.

Mayerhofer, M.J., Lolon, E.P., Youngblood, J.E., Heinze, J.R., 2006. Integration of microseismic fracture mapping results with numerical fracture network production modeling in the barnett shale. In: SPE Annual Technical Conference and Exhibition. OnePetro.

McKinnon, K.I., 1998. Convergence of the nelder–mead simplex method to a nonstationary point. SIAM J. Optim. 9, 148–158.

Mitchell, M., 1998. An Introduction to Genetic Algorithms. MIT Press.

Mualem, Y., 1976. A new model for predicting the hydraulic conductivity of unsaturated porous media. Water Resour. Res. 12, 513–522.

Nejadi, S., Trivedi, J.J., Leung, J., 2017. History matching and uncertainty quantification of discrete fracture network models in fractured reservoirs. J. Pet. Sci. Eng. 152, 21–32.

Neuman, S.P., 2005. Trends, prospects and challenges in quantifying flow and transport through fractured rocks. Hydrogeol. J. 13, 124–147.

Neuman, S.P., 2008. Multiscale relationships between fracture length, aperture, density and permeability. Geophys. Res. Lett. 35.

Pant, M., Zaheer, H., Garcia-Hernandez, L., Abraham, A., et al., 2020. Differential evolution: A review of more than two decades of research. Eng. Appl. Artif. Intell. 90, 103479.

Ping, J., Zhang, D., 2013. History matching of fracture distributions by ensemble Kalman filter combined with vector based level set parameterization. J. Pet. Sci. Eng. 108, 288–303.

Richards, L.A., 1931. Capillary conduction of liquids through porous mediums. Physics 1, 318–333.

Ringel, L.M., Jalali, M., Bayer, P., 2021. Stochastic inversion of three-dimensional discrete fracture network structure with hydraulic tomography. Water Resour. Res. 57, e2021WR030401.

Sicking, C., Malin, P., 2019. Fracture seismic: Mapping subsurface connectivity. Geosciences 9 (508).

Sivanandam, S., Deepa, S., 2008. Genetic algorithms. In: Introduction to Genetic Algorithms. Springer, pp. 15–37.

Tran, N.H., 2007. Simulated annealing technique in discrete fracture network inversion: optimizing the optimization. Comput. Geosci. 11, 249–260.

Van Genuchten, M.T., 1980. A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. Soil Sci. Am. J. 44, 892–898.

Willis, M.E., Burns, D.R., Rao, R., Minsley, B., Toksöz, M.N., Vetri, L., 2006. Spatial orientation and distribution of reservoir fractures from scattered seismic energy. Geophysics 71, O43–O51.

Witherspoon, P.A., Wang, J.S., Iwai, K., Gale, J.E., 1980. Validity of cubic law for fluid flow in a deformable rock fracture. Water Resour. Res. 16, 1016–1024.

Wood, A.T., 1994. Simulation of the von mises fisher distribution. Comm. Statist. Simulation Comput. 23, 157–164.

Wu, R., Chen, X., Hammond, G., Bisht, G., Song, X., Huang, M., Niu, G.Y., Ferre, T., 2021. Coupling surface flow with high-performance subsurface reactive flow and transport code pflotran. Environ. Model. Softw. 137, 104959.

Xiang, J., Yeh, T.C.J., Lee, C.H., Hsu, K.C., Wen, J.C., 2009. A simultaneous successive linear estimator and a guide for hydraulic tomography analysis. Water Resour. Res. 45.

Xu, C., Dowd, P., Wyborn, D., 2013. Optimisation of a stochastic rock fracture model using markov chain Monte Carlo simulation. Min. Technol. 122, 153–158.

Zha, Y., Yeh, T.C.J., Illman, W.A., Tanaka, T., Bruines, P., Onoe, H., Saegusa, H., 2015. What does hydraulic tomography tell us about fractured geological media? A field study and synthetic experiments. J. Hydrol. 531, 17–30.

Zhang, L., Cui, C., Ma, X., Sun, Z., Liu, F., Zhang, K., 2019. A fractal discrete fracture network model for history matching of naturally fractured reservoirs. Fractals 27, 1940008.