HatchFrac: A fast open-source DFN modeling software

Weiwei Zhu[1], Siarhei Khirevich[1], and Tad W Patzek[1]

[1]Ali I. Al-Naimi Petroleum Engineering Research Center (ANPERC)

November 26, 2022

Abstract

This paper introduces a comprehensive C++ software package, HatchFrac, for stochastic modelling of fracture networks in two and three dimensions. Two main methods, the inverse CDF method and acceptance-rejection method, are applied to generate random variables from the stochastic distributions commonly used in discrete fracture network (DFN) modelling. The multilayer per-ceptron (MLP) machine learning approach, combined with the inverse CDF method, is implemented to generate random variables following any sampling distribution. To make the code faster, we extend the Newman-Ziff to determine clusters in the fracture networks. When combined with the block method, the Ziff algorithm improves the coding efficiency significantly. The software generates the T-type fracture intersections in the network, which can be used in applications involving fracture growth or incorporating geomechanics. We introduce three applications of HatchFrac that demonstrate the versatility of our software: percolation analysis, fracture intensity analysis, and flow and connectivity analysis.

HatchFrac: A fast open-source DFN modeling software

Weiwei Zhu[a,*], Siarhei Khirevich[a], Tad W. Patzek[a]

[a] *Ali I. Al-Naimi Petroleum Engineering Research Center (ANPERC), King Abdullah University of Science and Technology, Jeddah, Mecca, KSA*

Abstract

This paper introduces a comprehensive C++ software package, HatchFrac, for stochastic modelling of fracture networks in two and three dimensions. Two main methods, the inverse CDF method and acceptance–rejection method, are applied to generate random variables from the stochastic distributions commonly used in discrete fracture network (DFN) modelling. The multilayer perceptron (MLP) machine learning approach, combined with the inverse CDF method, is implemented to generate random variables following any sampling distribution. To make the code faster, we extend the Newman–Ziff to determine clusters in the fracture networks. When combined with the block method, the Ziff algorithm improves the coding efficiency significantly. The software generates the T-type fracture intersections in the network, which can be used in applications involving fracture growth or incorporating geomechanics. We introduce three applications of HatchFrac that demonstrate the versatility of our software: percolation analysis, fracture intensity analysis, and flow and connectivity analysis.

1. Introduction

Fractures such as joints, faults, pressure solution seams, and deformation bands are ubiquitous in crustal rocks. Natural fractures usually comprise com-

*Corresponding author
Email address: weiwei.zhu@kaust.edu.sa (Weiwei Zhu)

plex networks, and they vary in size over scales ranging from microns to hundreds of kilometres [1, 2]. Throughout this wide range of scales, fracture networks dominate the geomechanical and hydrological behavior of subsurface rocks and play an essential role in many engineering fields, e.g., in hydrology, waste disposal, earthquake physics, and water, petroleum and geothermal reservoir exploitation [3, 4, 5, 6, 7, 8]. The fracture shapes are complex and irregular because of the anisotropic and heterogeneous characteristics of rocks and the complex geomechanical environments. Natural fractures have complex rough surfaces [9, 10]. The tortuosity of the flow paths in a fracture and the stress impact on fractures are also important for the flow in fractures[11]. Complex geometric shapes and dynamic variability of fractures make it very difficult to characterize fracture networks in detail. A practical alternative is the discrete fracture network (DFN) modelling method, where important geometrical and topological structures of fracture systems are preserved.

A "discrete fracture network" (DFN) refers to a computational model that explicitly represents the geometrical properties of individual fractures, mainly their orientations, sizes, positions, shapes, and apertures [12]. This modeling method was first applied to characterize and simulate flow and transport in natural fractures for the emerging high-level nuclear waste repository studies in the U.S. and Sweden in the 1970s and 1980s. Over the last four decades, DFN modeling has been extensively applied in different engineering fields. Bour and Davy [13], Robinson [14], Andresen et al. [15], Wilcock [16], De Dreuzy et al. [17], Baecher et al. [18] and many others have implemented the DFN approach to simulate fractures and fracture networks in two- or three-dimensions (2D or 3D) to investigate their percolation properties or topological structures. However, detailed information about how to construct the fracture network is usually not available from their papers. A few commercial software applications

can construct fracture networks in 2D and 3D, like FracMan from Golder Associates [19] or Petrel package from Schlumberger [20]. Because of their "closed source" strategy, end users cannot improve the underlying algorithms to fulfill specific research requirements. Alghalandis [21] developed open-source software in the Matlab environment. The powerful Matlab function libraries and toolboxes make the programming simpler. Nonetheless, as a high-level programming language, Matlab cannot deal with hundreds of thousands or millions of fractures, especially in 3D. Furthermore, specific functions, such as the cluster-checks, are hard to vectorize, and Matlab is slow in processing the "for loops" and "if statements."

In this paper, we present an efficient fracture network modelling package implemented in a C++ environment. The paper is organized as follows: Section 2 introduces basic concepts used in constructing a fracture network, including the fracture shapes, different stochastic distributions applied to describe fracture geometries, the intersection analysis, cluster analysis, and fracture growth analysis. Section 3 applies the software to percolation analysis, fracture density analysis, and flow and connectivity analysis. Appendix A provides advanced procedures to generate random variables following different stochastic distributions.

2. Basics of fracture networks

In this section we discuss basic concepts and algorithms for generating fracture networks in 2D and 3D.

2.1. Fracture shape

Complex geometric shapes and their dynamic variations make it almost impossible to characterize fracture networks in detail. Practically, a line segment is often used to represent a single fracture in 2D space [13, 14, 15, 21]. In 3D

space, several simple geometrical shapes are proposed to avoid excessive complexity. The Random Disk Model proposed by Baecher et al. [18] is widely adopted as the starting point due to its simplicity. The disk shape is applied in DFN modeling software, e.g., FracMan and rock mechanics software, e.g., Itasca. Elliptic, square or rectangular shapes are also commonly used in DFN modeling [16, 17, 21].

As Jing and Stephansson [22] pointed out, the significance of the fracture shape decreases with an increase in the fracture population size. In HatchFrac, we choose to use a random convex polygon with four vertices to represent a single fracture in 3D space. A random polygon preserves some degree of irregularity compared with a disk, and it can easily be converted to an ellipse or other polygon shapes by adding a few more vertexes and minor adjustments to the coordinates. Also, the intersection analysis of convex polygons is much easier than that for ellipses, which we discuss in the intersection analysis below.

2.2. Stochastic distributions of main fracture geometries

It is impossible to map the full extent of all fractures present in a subsurface formation in three dimensions. However, we can develop statistics on fracture orientations, intensities, apertures and lengths based on the measurements from outcrops or well-logs [23]. Constrained by these statistical properties, a stochastic fracture network can be constructed. The basic geometrical properties required to describe a single fracture are its shape (in 3D), length (in 2D), orientation, aperture, and position of the fracture center.

Different distributions are implemented to characterize the main geometric properties of the fracture network [24]. Exponential, gamma, log-normal and power-law distributions have been proposed to describe fracture lengths [25, 26, 27, 28, 29, 30]. Field observations and analog experiments suggest prevalence of power-law distribution [31, 32, 13, 33, 24]. Log-normal and power-law

distributions are used to describe the aperture variations [34, 35, 36]. Walmann et al. [37], Olson [38], Renshaw and Park [39], Bai et al. [40] found that there is a scaling relation between the aperture and length of fractures. The orientation of fractures is usually described by a von Mises–Fisher distribution [41, 42, 43]. From analyzing a collection of outcrop maps, we find that most fracture networks in outcrop maps have their concentration parameter $\kappa < 3$[23].

Two methods are commonly used to generate observations from a particular distribution in statistics. These are the inverse CDF method and the acceptance–rejection method[44]. Appendix A contains a detailed derivation on how to apply those methods to generate variables following the aforementioned stochastic distributions. C++ code is also available online (`https://data.mendeley.com/datasets/zhs97tsdry/1`).

Spatial distributions to characterize the positions of fracture centers are more complex. For simplicity, uniform spatial density distribution is commonly applied to describe the positions of fracture centers [13, 45, 46, 47, 48]. However, realistic fracture networks rarely have uniformly distributed centers. Darcel et al. [49] studied the connectivity of fracture networks with the fracture center positions following a fractal spatial density distribution that brings clustering effects and might be closer to reality[50, 23]. A multiplicative cascade process [51, 52, 53] is applied to generate a fractal spatial density distribution of fracture centers. In 2D, if the fractal dimension is 2, the fractal spatial density distribution reduces to a uniform distribution. If the dimension is smaller than 2, there will be fracture clustering. Similarly, in 3D, the corresponding limiting dimension for uniform fracture distribution is 3. Fig. 1 shows sketch maps of the fractal and uniform spatial density distributions in 2D and 3D, respectively.

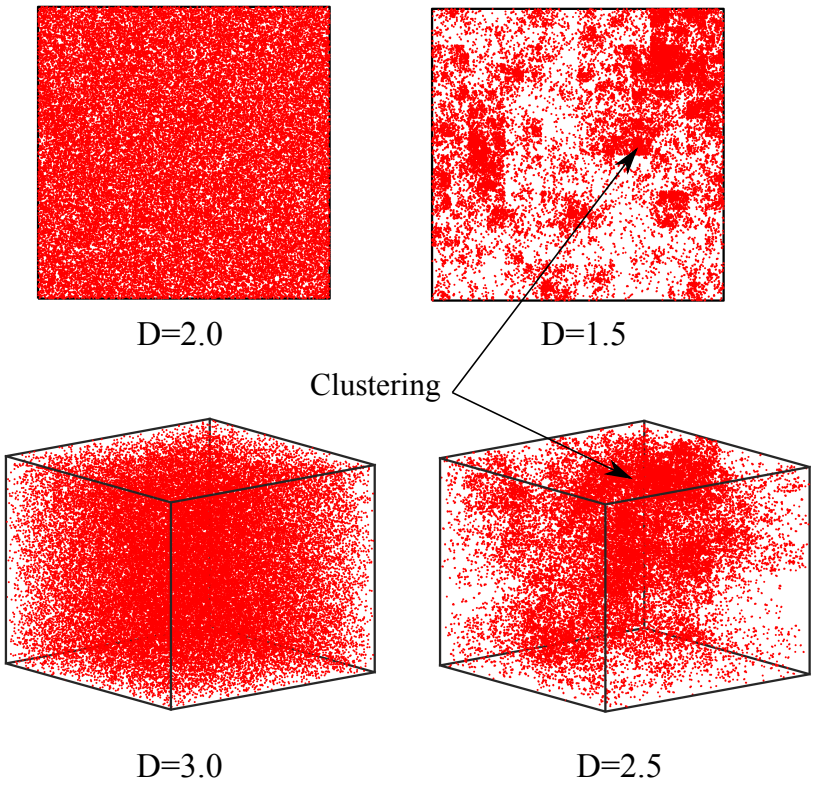

Figure 1: 50,000 2D/3D spatial points follow a uniform (**Left**) or a fractal spatial density distribution (**Right**) with the fractal dimension $D = 1.5/2.5$.

2.3. Machining learning for any sampling distribution

Continuous statistic distribution is an approximation of the sampling distribution from measurements. In some circumstances, a single continuous distribution is insufficient to fit the finite data, and we may want to generate random variables directly from a finite frequency histogram. Machine learning or artificial neural network [54, 55, 56] is a good option to do the regression and interpolation from finite samples. Multilayer perceptron (MLP) is a feedforward artificial neural network class that consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. This method is easy to implement and able to fit any sampling distribution within a given tolerance.

The backpropagation method is applied to train the data [57] with a nonlinear `Sigmoid` function as the activation function.

In this research, we recommend the MLP structure with five or six layers depending on the complexity of the histogram and four to six nodes in each hidden layer. The input nodes are fed with training data from measurements. Afterwards, the inverse CDF method is applied to generate a random variable following the sampling distribution. To sample from a distribution $\mathtt{p(x)}$, we can sample u uniformly on [0, 1] and calculate

$$x = \phi_{\mathtt{x}}^{-1}(u), \tag{1}$$

where $\mathtt{p(x)}$ and $\phi_{\mathtt{x}}^{-1}$ are the probability and inverse cumulative distribution function. The inverse of the cumulative distribution function $\phi_{\mathtt{x}}^{-1}$ is not available because of the unknown $\mathtt{p(x)}$ of the sampling distribution. Instead, we can implement a forward method to obtains x as the root of $\phi(\mathtt{x}) - u = 0$ with a numerical method (i.e., bisection method). The cumulative distribution function $\phi(\mathtt{x})$ is a monotonically increasing function that guarantees a unique solution for x. An example of generating random variables from a sampling distribution is shown in Fig. 2. By applying the MLP method, we can generate variables following any sampling distribution. The C++ code for an MLP algorithm can also be found online (`https://data.mendeley.com/datasets/zhs97tsdry/1`).

2.4. Intersection analysis

Connectivity is a fundamental feature of fracture networks, and an important measure for assessing flow transport through fractures [59]. Common methods adopted to investigate the connectivity of fracture networks include percolation theory [14, 45, 13, 60, 33], connectivity function method [61, 62, 63] and

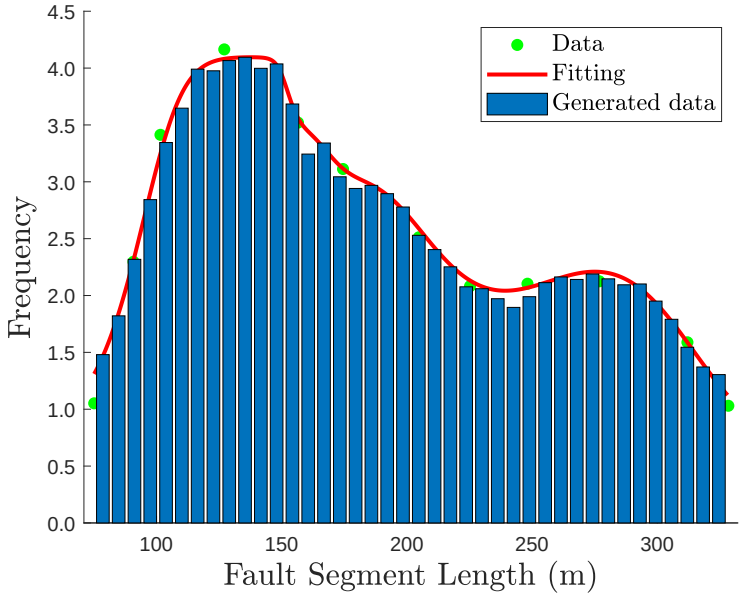

Figure 2: An example of generating random variables that follow a discrete sampling distribution. The green points are the frequency data of the fault segment lengths from de Joussineau and Aydin [58]'s paper. The red curve is the fit achieved with the MLP method. The blue bars are 10,000 data points generated through the inverse CDF method (the frequency is properly scaled down to fit the curve).

intersection relationship analysis [35, 64]. The intersections between fractures are essential to analyze connectivity of a fracture network. In 2D fracture networks, the function of checking intersections between two nonparallel fractures is straightforward, and two steps are sufficient.

1. Check the intersection of two lines on which two fractures lie and get the intersection point if these lines intersect.
2. Check whether the intersection point belongs to the two fractures simultaneously.

To check the intersection between nonparallel 3D fractures is more complex because of the irregular polygon geometry. We can resolve this problem into subproblems and check the intersection between a line and a plane. The pseudocode for checking intersections between two 3D fractures is listed below.

Algorithm 1: Intersection of 3D fractures

```
Data: Fracture A and B
Result: Check the intersection between two 3D fractures
begin
    for Each edge of Fracture A do
        Check the intersection between the line and the plane where the edge and
         Fracture B belong;
        Record the intersection point if they intersect;
    end
    for Each edge of Fracture B do
        Check the intersection between the line and the plane where the edge and
         Fracture A belong;
        Record the intersection point if they intersect;
    end
    if There is at least one intersection point belonging to both Fracture A and B
     then
        Return true;
    else
        Return false;
    end
end
```

The intersection functions for 2D and 3D fractures have the same complexity of $\mathcal{O}(1)$, while the 3D intersection function has a few more steps than the 2D one.

2.5. Efficient cluster analysis

A connected fracture network is the pathway of the fluid flow in low permeability formations. Therefore, a cluster-check algorithm is necessary to find clusters of intersecting fractures. The Hoshen–Kopelman algorithm [65] and its enhancements [66, 67] are widely used to check clusters in bond or site percolation problems, and in nonlattice environments. However, the Hoshen–Kopelman algorithm is a variation on the depth-first search and has a complexity of $\mathcal{O}(N^2 \ln N)$. This algorithm is inefficient in dealing with a large number of fractures. In 2001, Newman and Ziff proposed a fast Monte Carlo algorithm

[68]. Their algorithm can be implemented to check clusters in both bond or site percolation with a complexity of $\mathcal{O}(N)$.

We extended the Newman–Ziff algorithm to label clusters in 2D and 3D fracture networks and sped up the code. The intersection function needs to be implemented for all pairs of fractures, and it involves N^2 calls, where N is the number of fractures [46]. To further enhance the computational efficiency of the software, we divide the domain into smaller blocks with the size of B_s in 2D and 3D networks (the domain considered here is a square in 2D and a cube in 3D). Each fracture in the domain has an array to record the indices of blocks that fracture occupies. When we check the intersections for a given fracture, denoted as fracture A, only fractures that share the same blocks with fracture A should be checked for intersections. Most fractures in the domain are not checked for intersections, which saves a lot of computational time. The size of the block should be chosen wisely, because an unsuitable value can increase the computational time. It turns out selecting ten to twenty per cent of the system size as the block size yields good performance.

The fracture network is generated by adding fractures one by one until it fulfills a given stop criterion, such as reaching a predefined fracture intensity or a spanning cluster across the domain. Therefore, the cluster-check algorithm is employed whenever a new fracture is added. Each fracture has a value of "pointer to root" (`PTR`), and the default value is -1. If the `PTR` value is negative, it means that the corresponding fracture is the root fracture of a cluster, and the absolute value of `PTR` refers to the number of fractures in that cluster. If the `PTR` value is positive, the value of `PTR` points to the index of the root fracture. For example, if the n^{th} fracture A has a `PTR` value of -1, it means that fracture A is an isolated fracture and there is no fracture intersecting fracture A. If n^{th} fracture A has a `PTR` value of -15 and the m^{th} fracture B has a `PTR` value of n,

it indicates that fracture A is the root fracture of a cluster, and the cluster has 15 fractures directly or indirectly connected to fracture A. Fracture B is one of the fractures in the cluster because its `PTR` value is n (the index of fracture A). The pseudocode of the cluster-check algorithm that integrates the Newman–Ziff algorithm and block method is listed below:

Algorithm 2: Cluster-Check algorithm

Data: `Fractures`,`cFracture`, `FractureNum`, `Allblocknumber`,`Fractureblock`, `PTR`

Result: Check the cluster and label all fractures

begin

Step 1: Find all fractures that share at least one block with `cFracture`;

Step 2: Remove duplicate fractures in Step 1;

for *Each fracture i found in Step 2* **do**

Check the intersection between the fracture i and `cFracture`;

if *Fracture i intersects* `cFracture` **then**

Record the index of fracture i in an array, denoted as `IntersectionIndex`;

Record the length of `IntersectionIndex` as `countintersect`;

end

end

if `countintersect` $==$ *1* `/* Only one fracture intersects cFracture */`

then

`root` = `FindRoot`(`PTR`, `IntersectionIndex`[0]);

`PTR`[`FractureNum`] = `root`;

`PTR`[`root`] = `PTR`[`root`] - 1;`/* Add one fracture in this cluster */`

else if `countintersect` > 1 **then**

for *Each fracture i in* `IntersectionIndex` **do**

`root`[`i`] = `FindRoot`(`PTR`, `IntersectionIndex`[`i`]);

end

Remove duplicate roots in the array `root`;

Record the size of the array `root` as `countroot`;

for *Each root fracture i in the array* `root` **do**

`Num` = `Num` + `PTR`[`root`[`i`]];

`PTR`[`j`] = `root`[0];

for *Each fracture j in* `Fractures` **do**

if `PTR`[`j`] == `root`[`i`] **then**

`PTR`[`j`] = `root`[0];

`/* Merge all clusters into the first cluster */`

end

end

end

`PTR`[`FractureNum`] = `root`[0];

`PTR`[`root`[0]] = `Num` $-$ 1;

else

cFracture is an isolated fracture;

end

end

In the input argument, `Fractures` is a set of fractures including all previous fractures and the current fracture; `cFracture` is the current fracture; `FractureNum` is the index of the current fracture; `Allblocknubmer` is a matrix with $(\mathrm{L/Bs})^2$ rows in 2D and $(\mathrm{L/Bs})^3$ rows in 3D. Each row represents a block and records the indices of fractures in that block. `Fractureblock` is an array to record the indexes of blocks that the current fracture occupies; `PTR` is an array to record the `PTR` value of each fracture in `Fractures`. The `FindRoot(PTR, i)` is a recursive function used to find the index of the root fracture of the cluster where fracture i belongs. If fracture i itself is the root fracture, it will return the index of fracture i. The pseudocode of `FindRoot` is listed below:

Algorithm 3: FindRoot function

```
Data: PTR, i
Result: Find the root fracture of the cluster where fracture i belongs
begin
    if PTR[i] < 0 then
        return i;
    else
        return (FindRoot(PTR, PTR[i]));
    end
end
```

By combining the Newman–Ziff algorithm with the block method, the efficiency of the fracture cluster check algorithm is significantly improved. Fig. 3 shows the computational time of generating fractures and calls to the intersection function (tested on a PC: CPU one core,2.8 GHz, RAM 16 GB). The testing network has a system size of 100, and the block size is 20. The fractures have a constant length of 1, and uniformly distributed orientations and positions of fracture centers. It is worthwhile to notice that it is faster to generate 150,000 fractures in 3D rather than in 2D shown in Fig. 3 (a). With the same number of fractures and system size, 3D fractures are distributed in a volume compared with 2D fractures distributed in an area. The fracture intensity in

each 3D block is smaller, and fewer calls to the intersection function are needed in the 3D case, which reduces the computational time. Since we implement the same cluster-check algorithm in both 2D and 3D fracture networks, and the intersection functions have the same $\mathcal{O}(1)$ complexity in 2D and 3D, constructing 3D fracture networks is more memory-consuming, but not much more computationally expensive than 2D fracture networks. Fig. 3(b) reflects similar scaling slopes for both 2D and 3D fracture networks. Figs. 4 and 5 depict sketch maps

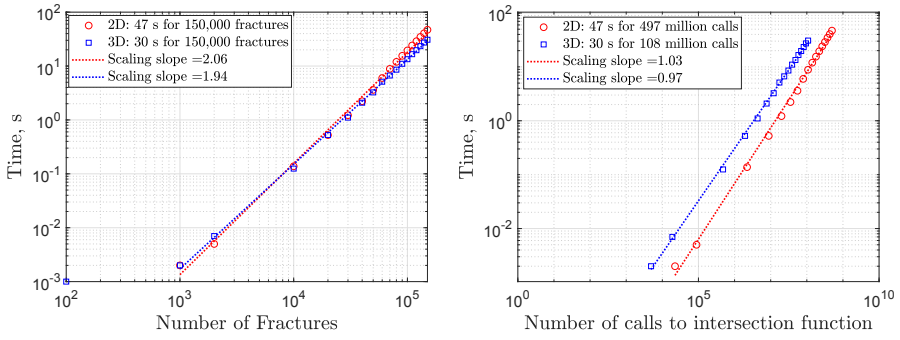

Figure 3: Left: computation time *vs* the number of fractures; Right: computation time *vs* the number of calls of the most time-consuming intersection function, in both 2D and 3D fracture networks. The timings include fracture and network generation, cluster checks and labelling.

of fracture networks in 2D and 3D spaces.

2.6. Fracture growth and T-type intersection

The processes above produce fractures exhibiting only X-type intersections (i.e., fractures cross one another) but not T-type intersections (i.e., one fracture terminates on another). T-type intersections are commonly observed in outcrops [23], and they help to enhance connectivity for a given fracture intensity because they reduce the number of dead-ends in the system [69]. To mimic the T-type intersections in the fracture network, the growth of fractures should be considered. Davy et al. [70, 71] modelled the fracture networks with T-type intersections and investigated fracture scaling characteristics. In Davy et al.

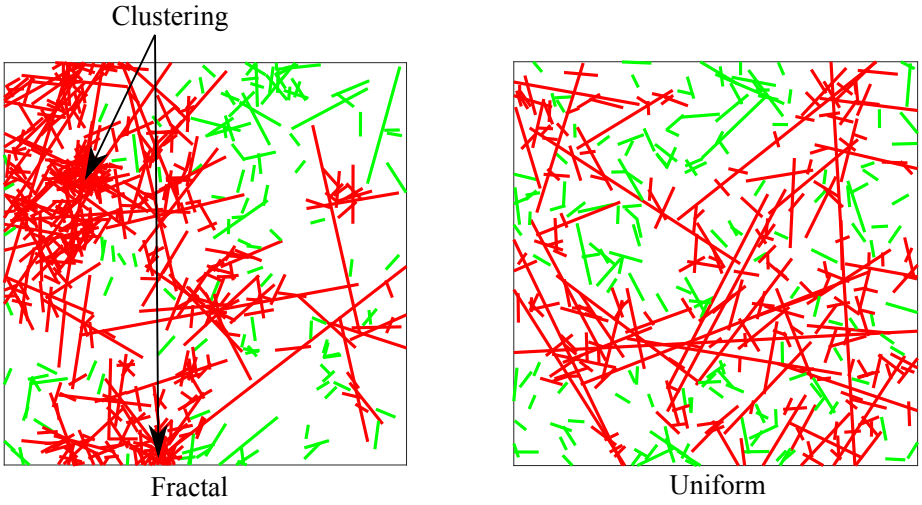

Figure 4: 2D fracture networks. The red line segments form the connected spanning cluster. The green line segments correspond to all other local connected clusters. In both networks, fracture orientations follow a uniform distribution, lengths obey a power-law distribution, and the fracture apertures are constant. The **left** network has fracture center positions that follow a fractal spatial density distribution with the fractal dimension of 1.5, and in the **right** network, the fracture centers follow a uniform distribution.

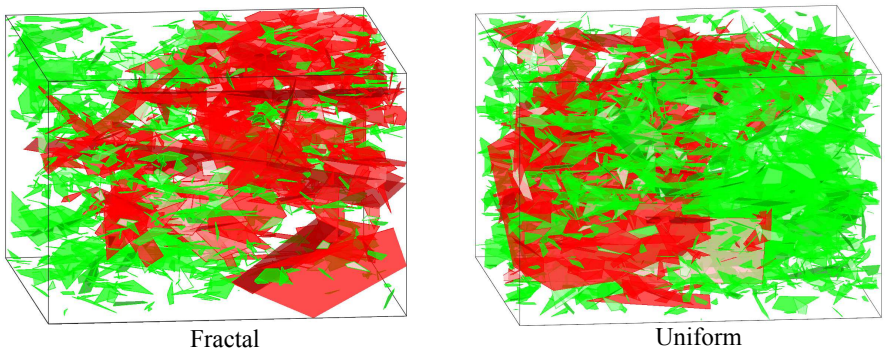

Figure 5: 3D fracture networks. The red polygons form the connected spanning cluster. The green polygons correspond to all other local connected clusters. In both networks, fracture orientations follow a uniform distribution, lengths obey a power-law distribution, and the fracture apertures are constant. The **left** network has fracture center positions that follow a fractal spatial density distribution with the fractal dimension of 2.5, and in the **right** network, the fracture centers follow a uniform distribution.

[71]'s paper, they discussed three steps to simulate the growth of fractures in 2D, which are nucleation, growth, and arrest. Similar concepts are adopted in

this paper to construct a fracture growth model in 2D and 3D spaces.

Preexisting depositional and mechanical weaknesses, such as crystal dislocation, grain boundaries, pores, microcracks, bedding planes, can reduce the tensile and shear strength of rocks and trigger tensile or shear fractures under applied stresses [72, 73]. The weaknesses that initiate fracture growth are called nuclei. The physics underlying the formation of nuclei, the rate of nucleation and the spatial and orientation distribution of nuclei is possibly related to stress condition and thermal activation [74, 75, 76]. To make the simulation practical, the nuclei are assumed to be uniformly distributed in both orientations and positions, and the nucleation rate is constant. It is straightforward to extend the spatial distribution and nucleation rate to a more realistic scenario if the nucleation mechanism can be stated in a specific mathematical format.

Fracture propagation in the subcritical regime is stable and quantifiable. The crack tip velocity is found to follow a power-law distribution [77, 78, 79, 80].

$$v = dl/dt = \mathrm{A}(\frac{\mathrm{K_I}}{\mathrm{K_{IC}}})^{n}, \tag{2}$$

where $\mathrm{K_I}$ is the opening-mode stress intensity factor at the fracture tip; $\mathrm{K_{IC}}$ is the opening-mode fracture toughness; A is the proportionality constant; and n is the subcritical fracture growth index, which varies widely depending on rock type and environmental conditions.

For the arrest criteria, it is reasonable to assume that large fractures inhibit the growth of smaller ones in their vicinity [31, 25], while the reverse is not likely to occur. The arrest condition in a 2D fracture network has two degrees of freedom. A fracture stops growing at a tip when it intersects the first large fracture at this tip. The other tip continues to grow until it intersects the second large fracture. For a 3D fracture network, the arrest criteria are more complex because of the random polygon shape. The growth of a 3D fracture is realized

by multiplying the coordinates with a scaling function, and the scaling factor is based on the velocity model. The scaling transformation is implemented on each non-intersecting vertex of the fracture. Once a vertex intersects a larger fracture, the vertex is replaced by the intersecting line segment and stops growing. We consider two modes of growth/stop models, which are shown in Fig. 6. In mode 1, the fracture stops growing when the two vertices on the longest diagonal line (BD) intersect large fractures (F_1 and F_2). In mode 2, the fracture stops growing when any three vertices of a fracture intersect larger fractures (F_1, F_2 and F_3). Figs. 7 and 8 illustrate fracture growth in 2D and 3D fracture networks. An explicit visualization of 3D fracture networks is difficult, and we only show the growth process of one fracture (green fracture) to illustrate the algorithm. The boundary plane is regarded as an infinitely large fracture, and the fracture tip stops growing when it intersects a boundary plane. The pseudocode of the

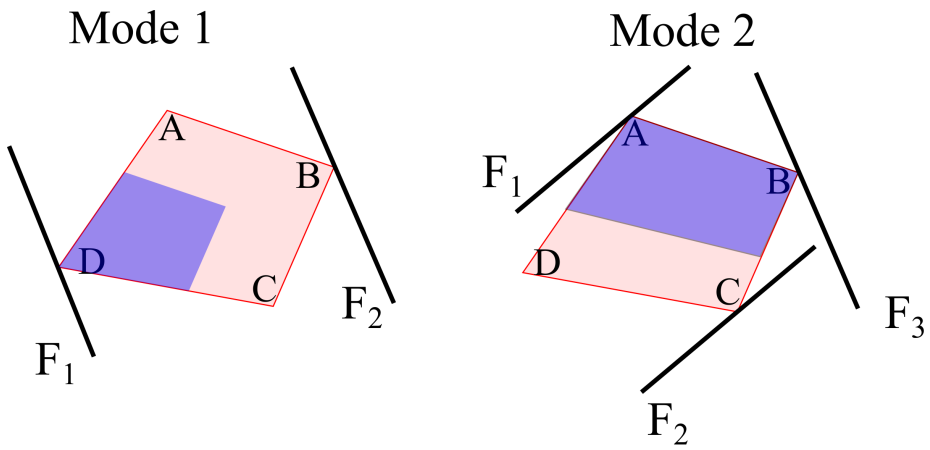

Figure 6: Illustration of two different arrest rules in 3D fracture networks. (**Left**) Mode 1. (**Right**) Mode 2. Fracture is modeled by a convex polygon with four vertices A, B, C, D. Fracture F_1, F_2, F_3 are fixed. Mode 1 and 2 depict the growing fracture in two different stages. Light purple represents the first stage where the fracture is still growing; light red represents the second stage corresponding to the point where the fracture stops growing. In mode 1, the fracture stops growing when the two vertices on the longer diagonal line (BD) intersect large fractures F_1 and F_2. In mode 2, the fracture stops growing when three vertices intersect large fractures F_1, F_2 and F_3.

growth process for a given number of nuclei is listed below.

Algorithm 4: Fracture growth algorithm

```
begin
    Initialization: Generate N nuclei;
    Initialization: t=0;
    while Not all N fractures are arrested do
        t=t+1;
        for int i = 0; i < N; i++ do
            if Fracture i is not arrested then
                Check intersections with all other fractures;
                /* Block method is applied to reduce computational time
                   here                                                  */
                if Arrest criteria fulfilled for Fracture i then
                    Fracture i is marked as arrested;
                else
                    Fracture i grows;
                    /* Only non-intersecting verticesgrow;              */
                    /* Intersecting verticesare replaced with the
                       intersection line segment and fixed;             */
                end
            end
        end
    end
end
```

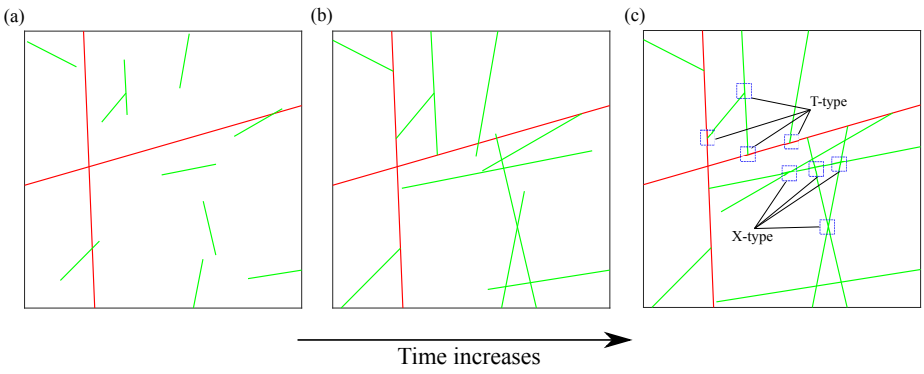

Figure 7: Illustration of fracture growth in a 2D fracture network. The sub-figure (c) is the last time step where all fractures stop growing. The T-type and X-type intersections are marked in the figure. Red line segments represent the preexisting old fractures. Green lines are the fractures growing from the initial nuclei.

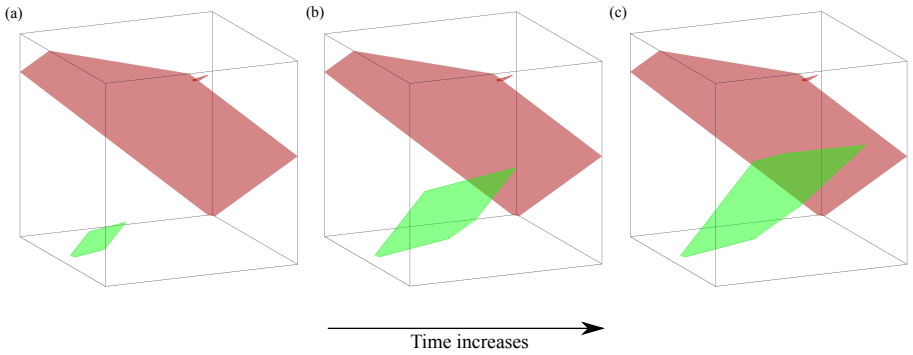

Figure 8: Illustration of fracture growth in a 3D fracture network. The sub-figure (c) is the last time step at which all fractures stop growing. Red polygons represent the preexisting old fractures. The green polygon is the fracture that grows from the initial nucleus. To better visualize the process, only a few red polygons and one green polygon are shown.

3. Applications

In this section, we present three applications of HatchFrac to demonstrate its utility. Each application is extended to a full research paper. A brief introduction and conclusions are presented here. The C++ code for generating 2D and 3D fracture networks and simulating the fracture growth process is available online (`https://data.mendeley.com/datasets/zhs97tsdry/1`).

3.1. Percolation analysis [81]

Percolation theory is widely used to analyze the connectivity of fracture networks. The percolation parameters commonly used to characterize fracture networks are the total excluded area A_{tex}, total self-determined area A_{tsd}, and the number of intersections per fracture I_{pf}. The formulas to calculate A_{tex}, A_{tsd}, I_{pf} in discrete fracture networks are listed in Eqs. (3) to (5).

$$A_{\text{tex}} = \frac{1}{(N-1)A} \sum_{i=1}^{N} \sum_{j=1 \neq i}^{N} L_i L_j \mid sin(\theta_i - \theta_j) \mid, \tag{3}$$

$$A_{\text{tsd}} = \frac{\sum_{i=1}^{N} L_i^2}{A}, \tag{4}$$

$$I_{\mathrm{pf}} = \frac{N_{In}}{N}, \tag{5}$$

where N is the total number of fractures in the fracture network, A is the area of the domain, L_i is the length of i^{th} fracture, θ_i is the orientation angle of i^{th} fracture, and N_{In} is the total number of intersections. These three quantities are percolation parameters for the constant-length fracture networks, but no one has investigated them in complex fracture networks. We investigate the variability of these three quantities in three types of fracture networks, in which fracture lengths follow a power-law distribution, fracture orientations follow a uniform distribution, and fracture center positions follow either a uniform distribution (type 1 and 2) or a fractal spatial density distribution (type 3). A sketch map of the three types of fracture networks is shown in Fig. 9.

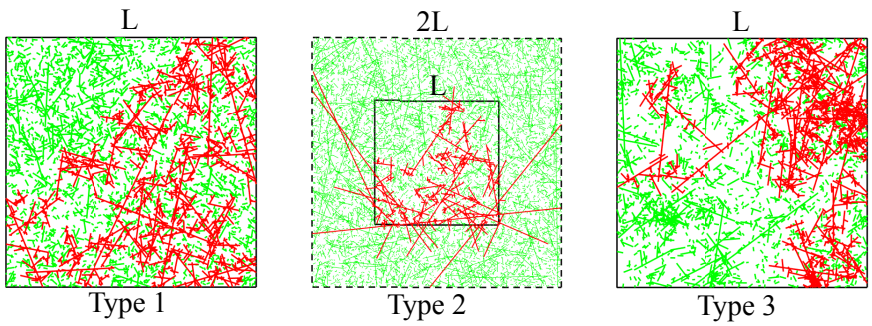

Figure 9: Sketch map of three types of fracture networks. The red line segments denote the connected spanning cluster, and the green line segments are not connected to the spanning cluster.

We show that in type 1 and type 2 fracture networks, these three quantities are percolation parameters only when the power-law exponent is larger than 3.5. In type 3 fracture networks, none of the three quantities is percolation parameters. We also investigated 18 outcrop fracture maps and found that the cm-scale and m-scale maps are closest to type 3 fracture networks. The outcrop

fractures cluster and have lengths that follow a power-law distribution with the exponent ranging from 2 to 3.

3.2. Fracture intensity analysis [82]

3D intensity parameters of fracture networks cannot be measured directly and are usually correlated with the lower dimensionality intensity parameters, such as P_{21}, P_{10}. Through generating 3D fracture networks and conducting 1D, 2D and 3D measurements on the networks, we performed a comprehensive correlation analysis between lower dimensionality measures, P_{10}, P_{20}, P_{21}, I_{2D} (total number of intersections per unit area) and higher dimensionality ones, P_{30}, P_{32}, I_{3D} (total number of intersections per unit volume). We also correlate cube samples and underlying fracture networks that represent cores or tunnels. The fracture networks are constrained by geomechanics principles and outcrop data to make them geologically meaningful. Four types of joints are generated, and the corresponding distributions are summarized in Table 1. A sketch map

Table 1: Distributions of each type of joints

Type of joints	Probability[a]	Center position	Strike	Dip	Length
1	0.02	Uniform[b]	von Mises–Fisher ($\mu = 90^\circ, \kappa = 300$)	90°	2L
2	0.02	Uniform	von Mises–Fisher ($\mu = 0^\circ, \kappa = 300$)	90°	Power-law[e] ($L_{max} = L, a = 3$)
3	0.72	Uniform	[d]N60°E, S60°E	90°	Power-law ($L_{max} = L, a = 2.5$)
4	0.24	Fractal[c]	Uniform ($[0, 2\pi]$)	Uniform ($[0, 2\pi]$)	Power-law ($L_{max} = L, a = 3$)

[a] probability of generation.
[b] uniform spatial distribution.
[c] fractal spatial density distribution with the fractal dimension of 2.5 in this research.
[d] dihedral angles equal to 60° and angle bisectors are parallel to σ_1.
[e] L_{max} is the maximum length of the fracture; a is the exponent of the power-law distribution.

of the 3D fracture network and the sampling methods is shown in Fig. 10.

We show that the orientation of fracture samples impacts correlations between the 2D and 3D parameters, and samples parallel to the principal stresses yield better correlations. 3D intensity parameters, P_{30}, I_{3D}, and P_{32} can be predicted from 2D or small cube samples. However, 1D intensity P_{10} does not

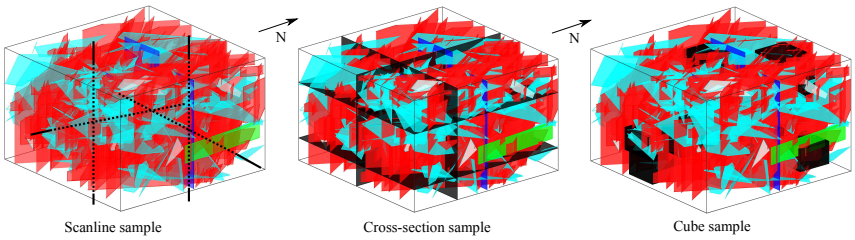

Figure 10: A sketch map illustrating different samples in the fracture network. The blue and green polygons represent the type 1 and type 2 tension joints. The red polygons represent the type 3 conjugate shear joints (microfaults). The cyan polygons represent the type 4 random shear joints (microfaults). The sampling lines, planes and cubes are black. The orientations of the maximum and minimum principal stress σ_1, σ_3 are north-south and east-west, respectively.

have a strong correlation with 3D intensity parameters. The size of the cube samples should be larger than 10 per cent of the original size to capture the main structural information. Furthermore, the minimum number of samples to reach a good correlation from 2D and cube samples are 20 and 60, respectively.

3.3. Flow and connectivity analysis [59]

In low permeability formations, connectivity of fractures determines the overall hydraulic diffusivity of the formation and measures the potential for fluid flow through their network [64, 13, 84, 85]. Through generating stochastic fracture networks and converting each fracture network to its graph representation, we utilize a topological concept—global efficiency—to evaluate the impact of geometry and topology of fractures networks on the connectivity. The main geometrical properties of the stochastic fracture networks considered include fracture lengths, orientations, apertures, positions of fracture centers. Six thousand different realizations have been generated to characterize these properties in each fracture network. Sketch maps of 2D and 3D fracture networks are shown in Figs. 4 and 5. The graph representations of 2D and 3D fracture networks depicted in Figs. 11 and 12. By removing the noncontributing nodes and links (dead-ends) in the graph, we preserve the relevant topological structure of

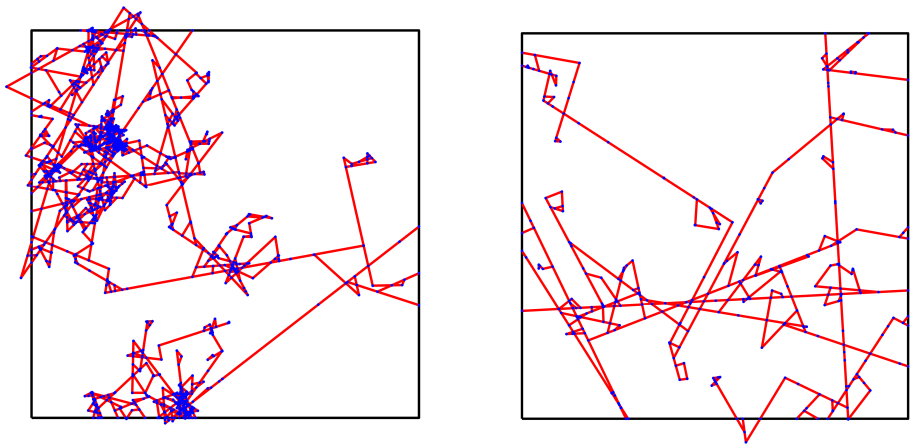

Figure 11: Graph representation of each fracture network in Fig.4. The blue points are the nodes including the start and end points of all fractures and all intersection points. The red line segments are the links between nodes.

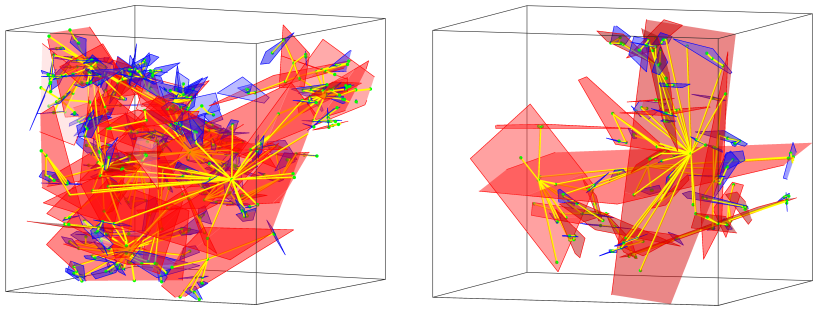

Figure 12: Graph representations of 3D fracture networks. The fractures in the **left** network follow a fractal spatial density distribution with the fractal dimension of 2.5. The **right** network has uniformly distributed fractures. The blue polygons are small fractures, and the red polygons are large fractures. The green points represent fracture centroids and intersection points. The yellow line segments are the links between the nodes.

the network, while reducing computational time significantly. Furthermore, it is more efficient to calculate flow using node-link formalism [86, 87, 88], rather than to solve it directly with the finite difference or finite element methods.

As a result, we find that the reduced fracture networks, consisting of the least resistant paths from inlet nodes (fractures) to all outlet nodes, contribute to the majority of fluid flow. Demonstration of the pressure head in reference

and reduced 2D and 3D fracture networks are shown in Fig. 13. We use them

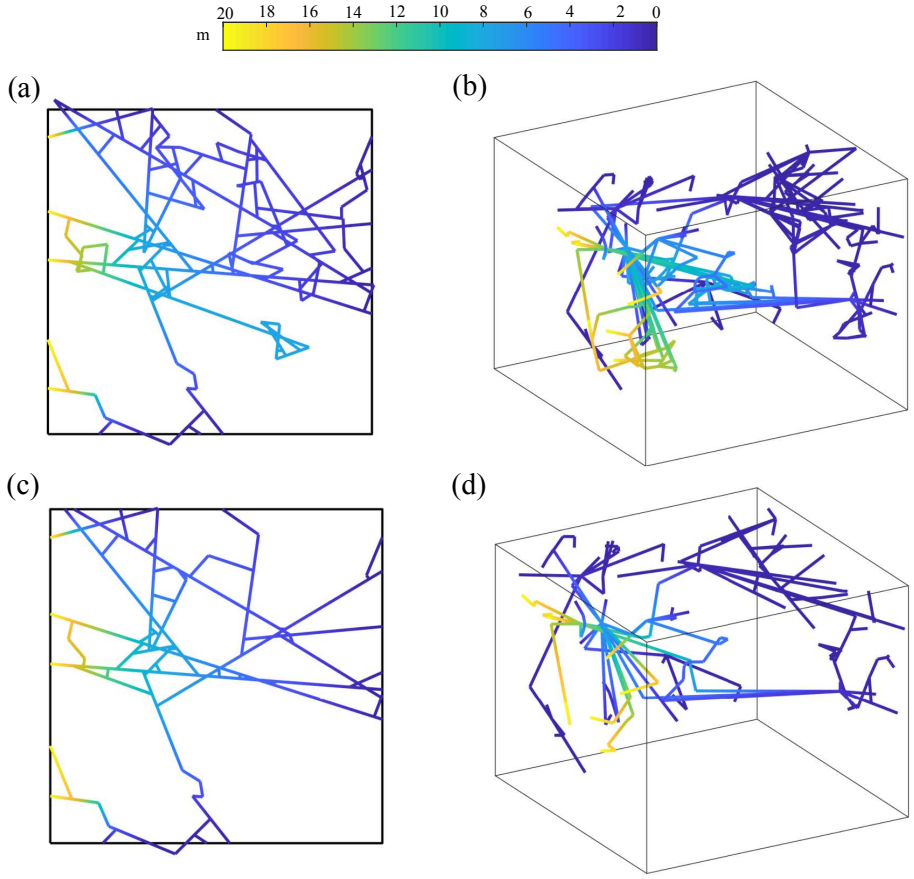

Figure 13: The hydraulic head distributions in the reference (a,b) and reduced (c,d) fracture networks. The constant pressure boundary condition is set on the domain, where the hydraulic head on the left edge in 2D and left face in 3D is 20 meters, and all other edges in 2D and faces in 3D have a hydraulic head of 0 meters.

355

356 to replace the original fracture networks and reduce computational time in most
357 cases. 3D fracture networks usually have higher global efficiency than 2D frac-
358 ture networks because they have better connectivity. All geometrical properties
359 impact the connectivity of a fracture system. Aperture distribution strongly
360 affects the global efficiency of a fracture network, and its influence is more sig-
361 nificant when large fractures dominate the system. Fracture clustering lowers
362 global efficiency in both 2D and 3D fracture networks. Global efficiency of 2D

and 3D fracture networks also decreases with the increasing exponent of the power-law distribution of fracture lengths, which means that the connectivity of the system decreases with an increasing number of small fractures. Realistic fracture networks, composed of several sets of fractures with constrained preferred orientations, share all the characteristics we have considered with the stochastic fracture networks in this work.

4. Conclusions

We detailed the procedures and algorithms of DFN modeling. In particular, we explained the choices of fracture shapes, the stochastic distributions that describe fracture geometries, the methods of generating random variables following given distributions, the intersection analysis, clustering analysis, and the fracture growth algorithm. We combined the MLP method with the inverse CDF method to generate random variables following any sampling distribution. By extending the Newman–Ziff algorithm to fracture networks and combining it with the block method, we significantly enhanced the efficiency of our software. Fracture growth algorithm can generate T-type intersections and can be further extended to investigate dynamic fracture growth problems that incorporate geomechanics. Three applications of the HatchFrac software in percolation analysis, intensity analysis, and flow and connectivity analysis were introduced to show the versatility of our software.

Acknowledgement

This project was supported by the baseline research funding from KAUST to Prof. Tad W. Patzek. The C++ code for generating random variables following different distributions and the C++ code for generating 2D and 3D fracture networks are available online (`https://data.mendeley.com/datasets/`

zhs97tsdry/1).

References

[1] M. H. Anders, S. E. Laubach, C. H. Scholz, Microfractures: A review, Journal of Structural Geology 69 (2014) 377–394.

[2] M. D. Zoback, S. M. Gorelick, Earthquake triggering and large-scale geologic storage of carbon dioxide, Proceedings of the National Academy of Sciences 109 (2012) 10164–10168.

[3] B. Berkowitz, Characterizing flow and transport in fractured geological media: A review, Advances in water resources 25 (2002) 861–884.

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

[5] J. Hyman, S. L. Painter, H. Viswanathan, N. Makedonska, S. Karra, Influence of injection mode on transport properties in kilometer-scale three-dimensional discrete fracture networks, Water Resources Research 51 (2015) 7289–7308.

[6] J. Hyman, G. Aldrich, H. Viswanathan, N. Makedonska, S. Karra, 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 Resources Research 52 (2016) 6472–6489.

[7] Y. Dong, Y. Fu, T.-C. J. Yeh, Y.-L. Wang, Y. Zha, L. Wang, Y. Hao, Equivalence of discrete fracture network and porous media models by hydraulic tomography, Water Resources Research 55 (2019) 3234–3247.

[8] W. Zhu, X. He, S. Khirevich, T. W. Patzek, Fracture sealing and its impact on the percolation of subsurface fracture networks, Earth and Space Science Open Archive (2021) 30. URL: https://doi.org/10.1002/essoar.10508231.1. doi:10.1002/essoar.10508231.1.

[9] R. Zimmerman, S. Kumar, G. Bodvarsson, Lubrication theory analysis of the permeability of rough-walled fractures, in: International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts, volume 28, Elsevier, 1991, pp. 325–331.

[10] X. He, H. Hoteit, M. AlSinan, H. Kwak, Modeling hydraulic response of rock fractures under effective normal stress, in: ARMA/DGS/SEG International Geomechanics Symposium, OnePetro, 2020.

[11] A. COOK, L. MYER, N. COOK, F. DOYLE, The effects of tortuosity on flow through a natural fracture, in: Rock mechanics contributions and challenges. US symposium. 31, 1990, pp. 371–378.

[12] Q. Lei, J.-P. Latham, C.-F. Tsang, The use of discrete fracture networks for modelling coupled geomechanical and hydrological behaviour of fractured rocks, Computers and Geotechnics 85 (2017) 151–176.

[13] O. Bour, P. Davy, Connectivity of random fault networks following a power law fault length distribution, Water Resources Research 33 (1997) 1567–1583.

[14] P. Robinson, Connectivity of fracture systems-a percolation theory approach, Journal of Physics A: Mathematical and General 16 (1983) 605.

[15] C. A. Andresen, A. Hansen, R. Le Goc, P. Davy, S. M. Hope, Topology of fracture networks, Frontiers in Physics 1 (2013) 7.

[16] P. Wilcock, The NAPSAC fracture network code, in: Developments in geotechnical engineering, volume 79, Elsevier, 1996, pp. 529–538.

[17] J.-R. De Dreuzy, P. Davy, O. Bour, Percolation parameter and percolation-threshold estimates for three-dimensional random ellipses with widely scattered distributions of eccentricity and size, Physical review E 62 (2000) 5948.

[18] G. Baecher, N. Lanney, H. Einstein, et al., Statistical description of rock properties and sampling, in: The 18th US Symposium on Rock Mechanics (USRMS), American Rock Mechanics Association, 1977.

[19] W. Dershowitz, G. Lee, J. Geier, T. Foxford, P. LaPointe, A. Thomas, Fracman Version 7.4—Interactive Discrete Feature Data Analysis, Geometric Modeling, and Exploration Simulation: User Documentation, 2014.

[20] J. J. Nehme, S. C. Srivastava, H. Bouzas, L. Carcasset, How Schlumberger achieved networked information leadership by transitioning to a product-platform software architecture, MIS Quarterly Executive 14 (2015) 105–124.

[21] Y. F. Alghalandis, ADFNE: Open source software for discrete fracture network engineering, two and three dimensional applications, Computers & Geosciences 102 (2017) 1–11.

[22] L. Jing, O. Stephansson, The Basics of Fracture System Characterization–Field Mapping and Stochastic Simulations, in: Developments in Geotechnical Engineering, volume 85, Elsevier, 2007, pp. 147–177.

[23] W. Zhu, X. He, R. K. Santoso?, G. Lei, T. Patzek, M. Wang, Enhancing fracture network characterization: A data-driven, outcrop-based analysis,

Earth and Space Science Open Archive (2021) 35. URL: https://doi.org/10.1002/essoar.10508232.1. doi:10.1002/essoar.10508232.1.

[24] E. Bonnet, O. Bour, N. E. Odling, P. Davy, I. Main, P. Cowie, B. Berkowitz, Scaling of fracture systems in geological media, Reviews of geophysics 39 (2001) 347–383.

[25] A. Nur, The origin of tensile fracture lineaments, Journal of Structural Geology 4 (1982) 31–40.

[26] P. A. Cowie, D. Sornette, C. Vanneste, Multifractal scaling properties of a growing fault population, Geophysical Journal International 122 (1995) 457–469.

[27] Y. Y. Kagan, Seismic moment distribution revisited: I. Statistical results, Geophysical Journal International 148 (2002) 520–541.

[28] I. Main, Statistical physics, seismogenesis, and seismic hazard, Reviews of Geophysics 34 (1996) 433–462.

[29] N. Odling, P. Gillespie, B. Bourgine, C. Castaing, J. Chiles, N. Christensen, E. Fillion, A. Genter, C. Olsen, L. Thrane, et al., Variations in fracture system geometry and their implications for fluid flow in fractures hydrocarbon reservoirs, Petroleum Geoscience 5 (1999) 373–384.

[30] S. Priest, J. Hudson, Estimation of discontinuity spacing and trace length using scanline surveys, in: International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts, volume 18, Elsevier, 1981, pp. 183–197.

[31] P. Segall, D. D. Pollard, Joint formation in granitic rock of the Sierra Nevada, Geological Society of America Bulletin 94 (1983) 563–575.

[32] A. Sornette, P. Davy, D. Sornette, Fault growth in brittle-ductile experiments and the mechanics of continental collisions, Journal of Geophysical Research: Solid Earth 98 (1993) 12111–12139.

[33] O. Bour, P. Davy, On the connectivity of three-dimensional fault networks, Water Resources Research 34 (1998) 2611–2622.

[34] J. Hooker, J. Gale, L. Gomez, S. Laubach, R. Marrett, R. Reed, Aperture-size scaling variations in a low-strain opening-mode fracture set, Cozzette Sandstone, Colorado, Journal of Structural Geology 31 (2009) 707–718.

[35] C. Barton, P. Hsieh, Physical and hydrologic-flow properties of fractures, in: 28th International Geological Congress Field Trip Guidebook, volume 385, 1989, p. 36.

[36] R. Marrett, O. J. Ortega, C. M. Kelsey, Extent of power-law scaling for natural fractures in rock, Geology 27 (1999) 799–802.

[37] T. Walmann, A. Malthe-Sørenssen, J. Feder, T. Jøssang, P. Meakin, H. Hardy, Scaling relations for the lengths and widths of fractures, Physical review letters 77 (1996) 5393.

[38] J. E. Olson, Sublinear scaling of fracture aperture versus length: an exception or the rule?, Journal of Geophysical Research: Solid Earth 108 (2003).

[39] C. Renshaw, J. Park, Effect of mechanical interactions on the scaling of fracture length and aperture, Nature 386 (1997) 482–484.

[40] T. Bai, D. D. Pollard, M. R. Gross, Mechanical prediction of fracture aperture in layered rocks, Journal of Geophysical Research: Solid Earth 105 (2000) 707–721.

[41] J.-J. Song, C.-I. Lee, M. Seto, Stability analysis of rock blocks around a tunnel using a statistical joint modeling technique, Tunnelling and underground space technology 16 (2001) 341–351.

[42] J. Kemeny, R. Post, Estimating three-dimensional rock discontinuity orientation from digital images of fracture traces, Computers & Geosciences 29 (2003) 65–77.

[43] A. E. Whitaker, T. Engelder, Characterizing stress fields in the upper crust using joint orientation distributions, Journal of Structural Geology 27 (2005) 1778–1787.

[44] L. Devroye, Nonuniform random variate generation, Handbooks in operations research and management science 13 (2006) 83–121.

[45] B. Berkowitz, Analysis of fracture network connectivity using percolation theory, Mathematical Geology 27 (1995) 467–483. doi:10.1007/BF02084422.

[46] O. Huseby, J. Thovert, P. Adler, Geometry and topology of fracture systems, Journal of Physics A: Mathematical and General 30 (1997) 1415.

[47] A. R. Piggott, Fractal relations for the diameter and trace length of disc-shaped fractures, Journal of Geophysical Research: Solid Earth 102 (1997) 18121–18125.

[48] B. Berkowitz, P. M. Adler, Stereological analysis of fracture network structure in geological formations, Journal of Geophysical Research: Solid Earth 103 (1998) 15339–15360.

[49] C. Darcel, O. Bour, P. Davy, J. De Dreuzy, Connectivity properties of two-dimensional fracture networks with stochastic fractal correlation, Water resources research 39 (2003).

[50] W. Zhu, S. Khirevich, T. Patzek, Percolation Properties of Stochastic Fracture Networks in 2D and Outcrop Fracture Maps, in: 80th EAGE Conference and Exhibition 2018, 2018.

[51] V. J. Martinez, B. J. Jones, R. Dominguez-Tenreiro, R. Weygaert, et al., Clustering paradigms and multifractal measures, Astrophysical Journal 357 (1990) 50.

[52] P. Meakin, Invasion percolation on substrates with correlated disorder, Physica A: Statistical Mechanics and its Applications 173 (1991) 305–324.

[53] P. Meakin, Diffusion-limited aggregation on multifractal lattices: A model for fluid-fluid displacement in porous media, Physical Review A 36 (1987) 2833.

[54] S. Haykin, Neural networks: a comprehensive foundation, Prentice Hall PTR, 1994.

[55] A. K. Jain, J. Mao, K. Mohiuddin, Artificial neural networks: A tutorial, Computer (1996) 31–44.

[56] J. M. Zurada, Introduction to artificial neural systems, volume 8, West publishing company St. Paul, 1992.

[57] J. Orbach, Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms., Archives of General Psychiatry 7 (1962) 218–219.

[58] G. de Joussineau, A. Aydin, Segmentation along strike-slip faults revisited, Pure and Applied Geophysics 166 (2009) 1575–1594.

[59] W. Zhu, S. Khirevich, T. W. Patzek, Impact of fracture geometry and topology on the connectivity and flow properties of stochastic fracture networks, Water Resources Research 57 (2021) e2020WR028652.

[60] M. Masihi, P. R. King, P. R. Nurafza, et al., Fast estimation of connectivity in fractured reservoirs using percolation theory, SPE Journal 12 (2007) 167–178.

[61] D. Allard, et al., On the connectivity of two random set models: the truncated Gaussian and the Boolean, in: Geostatistics Tróia'92, Springer, 1993, pp. 467–478.

[62] C. Xu, P. Dowd, K. Mardia, R. Fowell, A connectivity index for discrete fracture networks, Mathematical geology 38 (2006) 611–634.

[63] Y. F. Alghalandis, P. A. Dowd, C. Xu, Connectivity field: a measure for characterising fracture networks, Mathematical Geosciences 47 (2015) 63–83.

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

[65] J. Hoshen, R. Kopelman, Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm, Physical Review B 14 (1976) 3438.

[66] J. Hoshen, M. Berry, K. Minser, Percolation and cluster structure parameters: The enhanced Hoshen-Kopelman algorithm, Physical Review E 56 (1997) 1455.

[67] A. Al-Futaisi, T. W. Patzek, Extension of Hoshen–Kopelman algorithm to non-lattice environments, Physica A: Statistical Mechanics and its Applications 321 (2003) 665–678.

[68] M. E. Newman, R. M. Ziff, Fast Monte Carlo algorithm for site or bond percolation, Physical Review E 64 (2001) 016706.

[69] N. E. Odling, Scaling and connectivity of joint systems in sandstones from western Norway, Journal of Structural Geology 19 (1997) 1257–1271.

[70] P. Davy, R. Le Goc, C. Darcel, O. Bour, J.-R. De Dreuzy, R. Munier, A likely universal model of fracture scaling and its consequence for crustal hydromechanics, Journal of Geophysical Research: Solid Earth 115 (2010).

[71] P. Davy, R. Le Goc, C. Darcel, A model of fracture nucleation, growth and arrest, and consequences for fracture density and scaling, Journal of Geophysical Research: Solid Earth 118 (2013) 1393–1407.

[72] D. D. Pollard, A. Aydin, Progress in understanding jointing over the past century, Geological Society of America Bulletin 100 (1988) 1181–1204.

[73] J. C. Jaeger, N. G. Cook, R. Zimmerman, Fundamentals of rock mechanics, John Wiley & Sons, 2009.

[74] Z. Reches, D. A. Lockner, Nucleation and growth of faults in brittle rocks, Journal of Geophysical Research: Solid Earth 99 (1994) 18159–18173.

[75] V. Betekhtin, A. Kadomtsev, Evolution of microscopic cracks and pores in solids under loading, Physics of the solid state 47 (2005) 825–831.

[76] Y. Hamiel, O. Katz, V. Lyakhovsky, Z. Reches, Y. Fialko, Stable and unstable damage evolution in rocks with implications to fracturing of granite, Geophysical Journal International 167 (2006) 1005–1016.

[77] R. Charles, Dynamic fatigue of glass, Journal of Applied Physics 29 (1958) 1657–1662.

[78] J. E. Olson, Predicting fracture swarms—The influence of subcritical crack growth and the crack-tip process zone on joint spacing in rock, Geological Society, London, Special Publications 231 (2004) 73–88.

[79] M. Marder, J. Fineberg, How things break, Phys. Today 49 (1996) 24–29.

[80] T. Engelder, Tectonic implications drawn from differences in the surface morphology on two joint sets in the Appalachian Valley and Ridge, Virginia, Geology 32 (2004) 413–416.

[81] W. Zhu, S. Khirevich, T. Patzek, Percolation properties of stochastic fracture networks in 2d and outcrop fracture maps, in: 80th EAGE Conference and Exhibition 2018, volume 2018, European Association of Geoscientists & Engineers, 2018, pp. 1–5.

[82] W. Zhu, B. Yalcin, S. Khirevich, T. Patzek, Correlation analysis of fracture intensity descriptors with different dimensionality in a geomechanics-constrained 3d fracture network, in: Petroleum Geostatistics 2019, volume 2019, European Association of Geoscientists & Engineers, 2019, pp. 1–5.

[83] W. Zhu, B. Yalcin, S. Khirevich, T. Patzek, Correlation analysis of fracture intensity descriptors with different dimensionality in a geomechanics-constrained 3D fracture network, in: Fourth EAGE Conference on Petroleum Geostatistics, 2019.

[84] C. E. Renshaw, Connectivity of joint networks with power law length distributions, Water Resources Research 35 (1999) 2661–2670.

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

[86] T. W. Patzek, Verification of a Complete Pore Network Model of Drainage and Imbibition, Soc. of Petroleum Engineers J. 6 (2001) 144–156.

[87] Y. Fadakar-A, C. Xu, P. Dowd, Connectivity index and connectivity field towards fluid flow in fracture-based geothermal reservoirs, in: Proceedings of 38 Workshop on Geothermal Reservoir Engineering. Stanford University, Stanford, California, 2013, pp. 417–427.

[88] J. D. Hyman, A. Hagberg, G. Srinivasan, J. Mohd-Yusof, H. Viswanathan, Predictions of first passage times in sparse discrete fracture networks using graph-based reductions, Physical Review E 96 (2017) 013304.

[89] D. Kundu, R. D. Gupta, A convenient way of generating gamma random variables using generalized exponential distribution, Computational Statistics & Data Analysis 51 (2007) 2796–2802.

[90] L. Martino, D. Luengo, Extremely efficient generation of Gamma random variables for\alpha>= 1, arXiv preprint arXiv:1304.3800 (2013).

[91] P. Berens, et al., CircStat: a MATLAB toolbox for circular statistics, J Stat Softw 31 (2009) 1–21.

[92] G. Kurz, U. D. Hanebeck, Stochastic sampling of the hyperspherical von Mises–Fisher distribution without rejection methods, in: 2015 Sensor Data Fusion: Trends, Solutions, Applications (SDF), IEEE, 2015, pp. 1–6.

[93] G. Guennebaud, B. Jacob, et al., Eigen v3, http://eigen.tuxfamily.org, 2010.

Appendix A. Generating variables following different distributions

In this section, we introduce detailed procedures to generate random variables following a power-law, exponential, log-normal, gamma, and von Mises–Fisher distribution. Two main methods are introduced: the inverse CDF method and the acceptance–rejection method. The C++ code for generating these distributions are available online (`https://data.mendeley.com/datasets/zhs97tsdry/1`).

The inverse CDF method for generating a random sample is premised on the fact that a continuous cumulative distribution function, ϕ, is a one-to-one mapping of the CDF domain into the interval (0,1). Therefore, if u is a random variable uniformly distributed on (0,1), then $x = \phi^{-1}(u)$ has the distribution $p(x)$, where $p(x)$ is the corresponding probability distribution of ϕ. The inverse CDF method's key point is to calculate the inverse of the cumulative distribution function, which we can derive for power-law, exponential and log-normal distribution. In the following section, we derive the procedures to generate random variables following the three aforementioned distributions by applying the inverse CDF method. We also derive the truncated version for each of them since the real fracture parameters are finite and fall in a truncated range.

Appendix A.1. Power-law distribution (truncated)

If a random variable x ($x \geq 0$) follows a power-law distribution, the probability distribution function is

$$p(x) = \alpha x^{-a} \tag{A.1}$$

The cumulative distribution function is

$$\phi(x) = \int_0^x p(x)dx = \frac{\alpha}{1-a} x^{1-a} \tag{A.2}$$

Apply the inverse CDF method, which assumes the cumulative distribution function $\phi(x)$ is a random variable, u, uniformly distributed in [0,1], and we can get the random variable following a power-law distribution

$$x = \big(\frac{(1-a)}{\alpha}u\big)^{\frac{1}{1-a}} \tag{A.3}$$

If the random variable follows a truncated power-law distribution, which means the x_{min} and x_{max} are known, we have the following probability distribution function according to conditional probability

$$p(x \mid truncated) = \frac{p(x, truncated)}{p(truncated)} \tag{A.4}$$

Therefore, we obtain the cumulative distribution function of the truncated power-law distribution as

$$\phi(x \mid truncated) = \int_{x_{min}}^{x} p(x)dx \Big/ \int_{x_{min}}^{x_{max}} p(x)dx = \frac{\phi(x) - \phi(x_{min})}{\phi(x_{max}) - \phi(x_{min})} \tag{A.5}$$

Applying the idea of the inverse CDF method, in which $\phi(x \mid truncated)$ is a random variable, u_t, uniformly distributed on $[0, 1]$, we can have the random variable x follow a truncated power-law distribution.

$$x = \big(\frac{(1-a)}{\alpha}u\big)^{\frac{1}{1-a}} \tag{A.6}$$

where u is replaced with

$$u = \Big(\phi(x_{max}) - \phi(x_{min})\Big) \times u_t + \phi(x_{min}) \tag{A.7}$$

where $\phi(x)$ is shown in Eq. A.2.

Appendix A.2. Exponential law (truncated)

The same steps derived before can be applied in generating variables following an exponential distribution as well, which is listed hereafter:

1. The probability distribution of exponential distribution

$$p(x) = \lambda e^{-\lambda x} \tag{A.8}$$

2. The corresponding cumulative distribution function

$$\phi(x) = 1 - e^{-\lambda x} \tag{A.9}$$

3. Apply the inverse CDF method, and assume that u is a random variable uniformly distributed on [0,1].

$$x = \frac{\ln(1-u)}{-\lambda} \tag{A.10}$$

4. Replace u with Eq. A.11, to obtain a random variable following a truncated exponential distribution.

$$u = \Big(\phi(x_{max}) - \phi(x_{min})\Big) \times u_t + \phi(x_{min}) \tag{A.11}$$

where $\phi(x)$ is shown in Eq. A.9 and u_t is a random variable uniformly distributed on [0,1].

Appendix A.3. Log-normal distribution(truncated)

If a random variable x follows a log-normal distribution, which means $\ln(x)$ follows a normal distribution $N(\mu, \sigma^2)$. The same steps apply.

1. The probability distribution of log-normal distribution

$$p(x \mid \mu, \sigma^2) = \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-(\frac{\ln(x)-\mu}{\sqrt{2}\sigma})^2} \tag{A.12}$$

2. The corresponding cumulative distribution function

$$\phi(x) = \int_{-\infty}^{x} p(x \mid \mu, \sigma^2) = \frac{1}{2}[1 + \mathrm{erf}(\frac{\ln(x)-\mu}{\sigma\sqrt{2}})] \tag{A.13}$$

where erf() is the error function

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{A.14}$$

3. Apply the inverse CDF method, and assume that u is a random variable uniformly distributed on [0,1].

$$x = \exp\left(\mathrm{erf}^{-1}(2 \times u - 1) \times \sigma\sqrt{2} + \mu\right) \tag{A.15}$$

where $\mathrm{erf}^{-1}()$ is the inverse function of the error function.

4. Replace u with Eq. A.16, we can have a random variable following a truncated log-normal distribution.

$$u = \Big(\phi(x_{max}) - \phi(x_{min})\Big) \times u_t + \phi(x_{min}) \tag{A.16}$$

where $\phi(x)$ is shown in Eq. A.13 and u_t is a random variable uniformly distributed on [0,1].

Note that the expectation and variance of a log-normal distribution are different from μ and σ^2.

Instead, the mean is

$$E(x) = e^{\mu + \frac{\sigma^2}{2}} \tag{A.17}$$

and the variance is

$$V(x) = (e^{\sigma^2} - 1)(e^{2\mu+\sigma^2}) \tag{A.18}$$

Therefore, if the random variable x has a mean and variance A and B respectively, the corresponding μ and σ^2 in a log-normal distribution are

$$\sigma^2 = \ln(e^{\ln(B)-2\ln(A)} + 1) = \ln(\frac{B}{A^2} + 1) \tag{A.19}$$

$$\mu = \ln(A) - \frac{1}{2}\sigma^2 \tag{A.20}$$

For a gamma distribution and von Mises–Fisher distribution discussed below, the inverse of the cumulative distribution function is difficult to obtain, and the inverse CDF method is not applicable. The acceptance–rejection method is effective in dealing with this complex situation. The logic behind the acceptance–rejection method is to find a simpler distribution, $s(x)$, if the original distribution $p(x)$ is too complex and ensure that $s(x) > p(x)$. Then we generate a random variable x' following the simper distribution $s(x)$ and a random number u uniformly distributed on [0,1]. If $u \leq p(x')/s(x')$, accept $x = x'$, otherwise reject x' and regenerate x' and u. The key for the acceptance–rejection method is to find a proper distribution $s(x)$ that is close to $p(x)$ so that the acceptance rate will be high and the method will be efficient. The optimal distribution function is the supremum function of $p(x)$ theoretically. However, it is difficult to achieve the supremum function in most cases. Generating random variables following a gamma distribution and von Mises-Fisher distribution itself is a research problem. We are not going to propose new methods to realize the generation. Instead, we will introduce a few efficient and stable methods and provide the C++ program for them (`https://data.mendeley.com/datasets/zhs97tsdry/1`).

Appendix A.4. Gamma distribution

The choice of a suitable distribution function $s(x)$ for the gamma law distribution is nontrivial. If a random variable follows a gamma law distribution, the corresponding probability distribution function is

$$p(x \mid \alpha, \beta) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)} \tag{A.21}$$

where α is a shape parameter and β is a rate parameter, and its inverse is a scale parameter. The gamma distribution has a scaling characteristic, which means if x follows a gamma distribution, $x \sim \Gamma(\alpha, \beta)$, then cx also follows a gamma distribution with a rate factor equal to β/c, $cx \sim \Gamma(\alpha, \beta/c)$. Therefore, we can always generate a random variable following a $\Gamma(\alpha, 1)$ and then multiply the variable with $1/\beta$ to make the variable follow the distribution $\Gamma(\alpha, \beta)$. When the shape parameter $\alpha \leq 1$, we adopt the method proposed by Kundu and Gupta [89]. It has a lower rejection probability than the popular Ahrens-Dieter or Best method. For $\alpha > 1$, we adopt the approach proposed by Martino and Luengo [90], which uses another gamma density as the replacement distribution. It turns out to be simple and extremely efficient. Interested readers can find the details of the method in the papers mentioned, and the program is available online.

Appendix A.5. von Mises–Fisher distribution

If a random D-dimensional vector $\vec{x}$ follows a von Mises–Fisher distribution, the corresponding probability distribution function is:

$$p(\vec{x} \mid \vec{\mu}, \kappa) = C_D \exp(\kappa \vec{\mu}^T \vec{x}) \tag{A.22}$$

where $C_D(\kappa)$ is

$$C_D(\kappa) = \frac{\kappa^{D/2-1}}{2\pi^{D/2} I_{D/2-1}(\kappa)} \tag{A.23}$$

where I_ν denotes the modified Bessel function of the first kind of the order ν. The parameters $\vec{\mu}$ and κ are the mean direction and concentration parameters, respectively. κ controls the concentration degree of the distribution around the mean direction $\vec{\mu}$. When $\kappa = 0$, the von Mises–Fisher distribution degenerates to a uniform distribution. When κ is large, the distribution becomes very concentrated around the angle $\vec{\mu}$.

In our software, we only consider the vector $\vec{x}$ in 2D or 3D spaces. In two dimensional cases, the distribution becomes von Mises distribution, which is a probability distribution on the unit circle. When κ is large, the distribution is close to a normal distribution, and $1/\kappa$ is analogous to σ^2. We adopted the program proposed by Berens et al. [91], which is the algorithm used in the MATLAB toolbox, CircStat. In three dimensional cases, this distribution is also called the Fisher distribution and is a probability distribution on the unit sphere. We adopt the method proposed by Kurz and Hanebeck [92], which can be used to generate von Mises–Fisher distribution for any number of dimensions. However, we consider the special case of $D = 3$, where we can use the inverse CDF method instead of the acceptance–rejection method. Since the inverse CDF method is analytical, it is much more efficient than the acceptance–rejection method. Interested readers can find the details of the method in the papers mentioned, and the program is available online. To obtain the rotation matrix concerning the default mean direction, a C++ template library specialized for linear algebra, Eigen[93], is used to implement the `QR` decomposition. An example of von Mises–Fisher distribution on the unit sphere with different values of κ is shown in Fig. A.14.

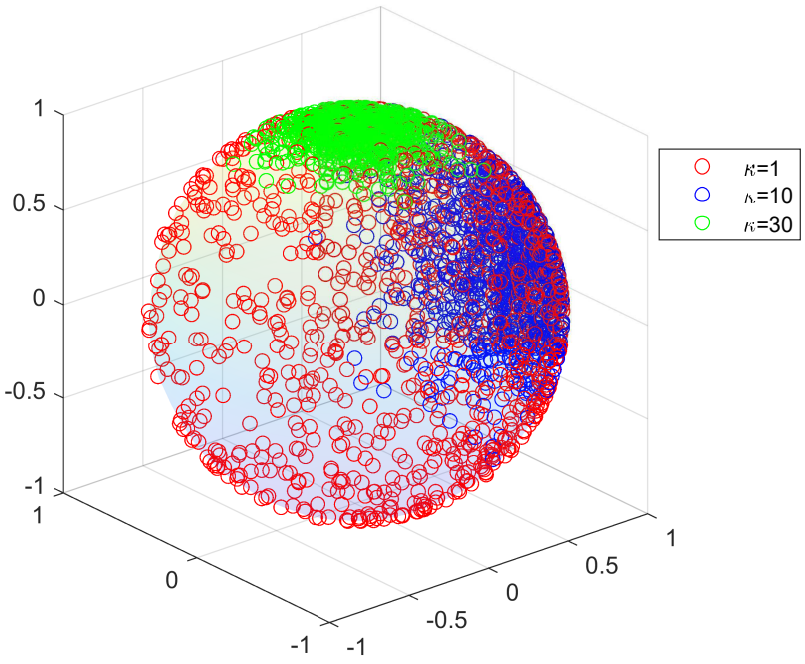

Figure A.14: Illustration of the von Mises–Fisher distribution on a unit sphere. The mean direction of the red and blue dots is (1,0,0), and the mean direction of the green dots is (0,0,1). κ controls the concentration degree of the distribution. The larger the κ, the more concentrated the distribution is.