PERGAMON

Computers & Geosciences 29 (2003) 107–111

COMPUTERS & GEOSCIENCES

www.elsevier.com/locate/cageo

Short note

Efficient algorithms for the determination of the connected fracture network and the solution to the steady-state flow equation in fracture networks

J.-R. de Dreuzy[a,*], J. Erhel[b]

[a] *Géosciences Rennes (UMR 6118) Campus de Beaulieu, 35042 Rennes Cédex, France*
[b] *IRISA/INRIA (UMR) Campus de Beaulieu, 35042 Rennes Cédex, France*

Received 7 December 2001; received in revised form 16 July 2002; accepted 19 July 2002

1. Introduction

In the last few years, the study of the possible underground storage of high level nuclear wastes has reinforced the need for an accurate modeling of the role of fractures on flow and transport in a broad range of geological media, in which fractures have been observed. Homogenization and continuous approaches are the most frequently used. Such approaches model the heterogeneity caused by the presence of fractures by homogeneous equivalent media. However they underestimate the effect of the heterogeneity among which the observed scale increase of permeability and dispersivity in natural fractured media (Clauser, 1992; Gelhar et al., 1992). The scale effects must be known accurately as they are at the heart of the modeling process of inferring field scale characteristics from laboratory measurements.

Alternatively another approach, the discrete approach, has been developed (Snow, 1969). Fractures are modeled by discontinuities (segments in 2D and planar objects in 3D) having different physical properties from the surrounding unfractured media. Integrating progressively the broad variety of fractures, discrete models become closer to physical processes and field observations but increasingly complex and computationally demanding (de Dreuzy et al., 2001). The broad variety of fractures comes especially from their widely scattered fracture length distribution, generally modeled by a power law without characteristic length scale apart from its endmost limits (de Dreuzy et al., 2001):

$$n(l) \sim l^a \quad \text{for } l \in [l_{\min}, l_{\max}], \tag{1}$$

where $l_{\min}$ and $l_{\max}$ are the minimal and maximal fracture lengths, respectively. Fig. 1A shows a typical 2D fracture network model having a power-law fracture length distribution.

The multi-scale nature of the fracture networks requires large networks to be simulated with a fine resolution. Moreover, the lack of field data makes necessary the use of a stochastic model and of Monte-Carlo simulations. As a result, the discrete approach must cope with the simulation of the hydraulic phenomena in a large number of large fracture networks made up often of several hundreds of thousands of fractures. Limitations in the modeling capabilities arise often from the execution time of the numerical models.

We show in this paper two algorithms for the two first modeling steps, which are the determination of the connected part of the network and the computation of the solution of the steady-state flow equation, that are orders of magnitude faster than the classically used algorithms. These two algorithms have been tested on bi-dimensional networks. We discuss briefly their extension to 3D networks. Although simple and straightforward to implement, these algorithms have not been described in previous reports, to the best of our knowledge.

2. Determination of the connected cluster

Although apparently simple, the determination of the connected cluster (i.e. the ensemble of fractures connected to the limits of the system) may be time consuming. Its limiting part is the search for the intersections between fractures. Once the fracture

*Corresponding author.
E-mail address: jean-raynald.de-dreuzy@univ-rennes1.fr (J.-R. de Dreuzy).

0098-3004/02/$ - see front matter
PII: S0098-3004(02)00081-X

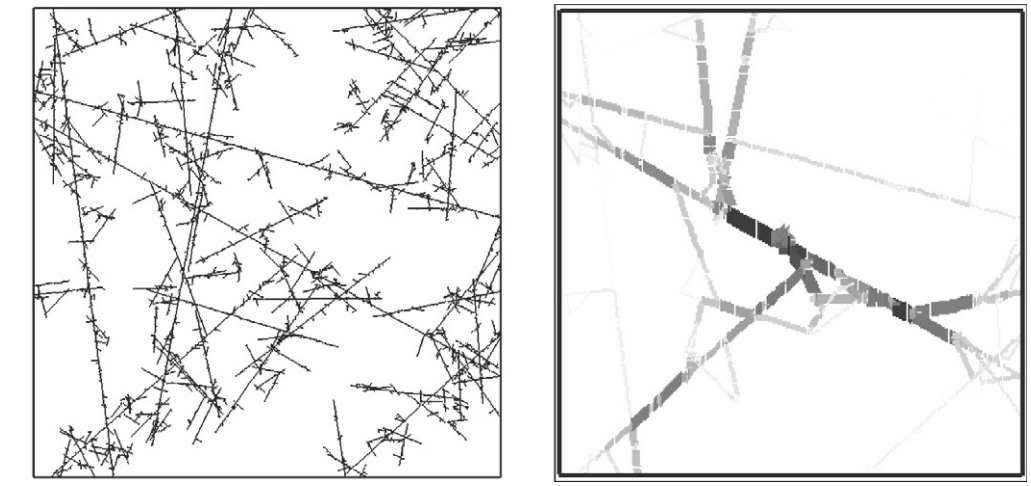

Fig. 1. (A) Model of fracture network having a power-law length distribution such as Eq. (2) with $a = 2.5$. Minimal fracture length is $l_{\min}$, global size of system L is equal to $100l_{\min}$ and maximal fracture length $l_{\max}$ is much larger than L. (B) Flow in steady state in network shown in (A). Boundary conditions are a fixed head on sides of system and an imposed flow on central node. Gray color and width of segments are proportional to flow.

intersections are determined, organizing n fractures in clusters either connected or isolated from the edges of the systems is an algorithm of complexity $O(n_s)$, where n_s is the number of segments of the network relating two consecutive points, the points of the network being the fracture ends and intersections. Indeed, the algorithm is the determination of the connected components of the graph whose vertices are the network segments. This algorithm is classical in graph theory and can be found for example in Gondran and Minioux (1984, p. 14). Once the graph organized in connected components, each component is scanned to test if it is connected to the boundary of the systems, which is an algorithm of complexity $O(n_p)$, where n_p is the number of network points. The connected components that are connected to the boundaries are retained and formed the "infinite cluster" (Stauffer and Aharony, 1992) in which flow takes place.

The classical search for intersections of a network made up of n fractures is the straightforward algorithm of complexity n^2 that tests all couples of fractures. We first describe and secondly study the complexity of a faster algorithm that we call the "scanning algorithm" adapted from computational geometry algorithms (Preparata and Shamos, 1985). The "scanning algorithm" consists in scanning the intersections of the system from one side of the system to the other one as illustrated in Fig. 2. The algorithm is made up of two successive steps. In the first step, the ends of the fractures are ordered by increasing x-coordinates (for example) in a vector. Fractures are characterized by the vertical band that they cover (Fig. 2). Fractures intersect only if their vertical band overlap. The second step thus consists in testing only the fractures whose vertical bands overlap.

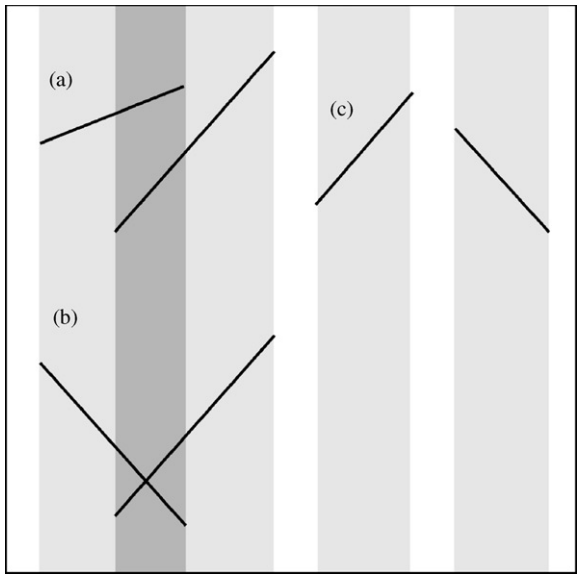

Fig. 2. Illustration of intersection tests by scanning method. In configurations (a) and (b), intersection is carried out because fractures cut same vertical zone underlined in dark gray. On other hand, in configuration (c), with fractures in different vertical zones, intersection between fractures is not tested.

The ordering algorithm for n fractures is of typical complexity $n \log n$. For each fracture, the number of intersection tests is proportional to the number of fractures intersecting the vertical band covered by the fracture. The width of the vertical band is of the order of the fracture length l and the average number of fractures per unit length is equal to n/L where L is the system site. By integrating over all fractures and by taking the mean fracture length $\langle l \rangle$, the number of intersection tests for all fractures is of the order of $n^2 \langle l \rangle / L$. The intersection tests of complexity n^2 remain thus the limiting factor over the ordering algorithm of order $n \log n$. The scanning algorithm of complexity $n^2 \langle l \rangle / L$ is faster than the classical search algorithm of complexity n^2 by the factor $L/\langle l \rangle$. When a is larger than 2, which is the most current case (Bonnet et al., 2001), the average fracture length $\langle l \rangle$ scales as the minimal fracture length $l_{\min}$ (Bour and Davy, 1997), which is orders of magnitudes smaller than the system size L. The scanning algorithm is expected to be orders of magnitude faster than the classical algorithm. We have compared both algorithms for a network made up of 200 000 fractures having a size ratio $L/l_{\min} = 1000$ and generated with the power-law fracture-length distribution of Eq. (1) characterized by the exponent $a = 2.5$. On a standard workstation, the determination of the fracture intersections took 3 h with the classical algorithm and 42 s with the scanning algorithm. The scanning algorithm is around 250 times faster than the classical algorithm, which is lower than the $L/l_{\min} = 1000$ times expected, but still far more efficient.

The scanning algorithm was accelerated by using the cross-product test (Cormen et al., 1990), which makes it possible to distinguish the configurations (a) and (b) from Fig. 2 without seeking the intersections explicitly. The cross product of two vectors u and v, noted $u \times v$, is a vector whose algebraic value is equal to $|u| \cdot |v| \cdot \sin(\theta)$, where θ is the angle between u and v. If the two vectors u and v are beginning at the same origin, a positive algebraic value of the cross product means that v is obtained from u by a rotation having a positive angle whereas if the algebraic value of the cross product is negative, it means that the rotation angle is negative. Provided that two segments M_1M_2 and N_1N_2 cross overlapping vertical bands, they intersect if and only if the algebraic value of their cross products $M_1M_2 \times M_1N_1$ and $M_1M_2 \times M_1N_2$ have opposite signs, which means that the two points N_1 and N_2 are on each other side of the segment M_1M_2.

The scanning algorithm can be extended to 3D fracture networks with the same complexity rules as those derived for 2D networks. We have verified that, as for the bi-dimensional networks, the scanning algorithm is much faster than the classical algorithm for 3D networks of disks. For example, for a network made up of 50 000 disks and of size $L/l_{\min} = 50$, the classical algorithm took 14 min whereas the scanning algorithm took 22 s, i.e. 40 times less, whereas the expected gain of time is 50 times.

3. Numerical solution of the steady-state flow equation

The flow equation in steady state is (de Marsily, 1986)

$$\nabla(K(\vec{r})\nabla h) = Q(\vec{r}), \tag{2}$$

where $\vec{r}$ is the position vector, h is the hydraulic head, Q is the source term, and K is the fracture permeability. The flow domain in which we solve Eq. (2) is the irregular grid made up of the fractures such as the one shown in Fig. 1A. The fracture properties intervene through the permeability K, which will be considered here as constant within a fracture and variable from fracture to fracture. We set the boundary conditions to mimic a well test made at constant flow rate Q. The flow Q is injected on the closest fracture to the center and a fixed head is given to the end points of the fractures located on the system edges. The flow corresponding to the network of Fig. 1A are presented on Fig. 1B.

Because of the medium complexity, Eq. (2) can only be solved numerically. We show how the heterogeneities of the medium intervene in the choice of the most efficient numerical method. We underline that our goal is to compare existing numerical methods in order to find efficient, precise and flexible tools of simulation. We thus used the free software PETSc[1] and UMFPACK[2] specialized in sparse linear systems, to perform and compare different algorithms. Matrices where stored in a sparse format implemented within the software used. This approach of using existing software allows to have efficient and validated tools of simulation with a minimum cost and time investment.

We discretize Eq. (2) on the irregular grid formed by the fracture network. Each node of the grid is either the intersection of two fractures or the end of a fracture. The discretization leads to a linear system of equations $A \cdot h = b$ of size equal to the number n_p of nodes of the network. The coefficients of the matrix A and of the second term b depend both on the configuration of the network and on the fracture permeabilities:

[1] The Portable, Extensible Toolkit for Scientific Computation, PETSc home page, 2001, http://www.mcs.anl.gov/petsc.

[2] A package implementing the Unsymmetric MultiFrontal method, UMFPACK home page, 2001, http://www.cise.ufl.edu/research/sparse/umfpack/.

$$
\begin{aligned}
&\text{if } i \text{ and } j \text{ are connected } (i \neq j) \text{ and } i \notin \Lambda \text{ and } j \notin \Lambda \\
&\quad A(i,j) = k_{ij}/d_{ij} \\
&\text{otherwise} \\
&\quad A(i,j) = 0 \\
&\text{if } i \notin \Lambda \\
&\quad A(i,i) = \textstyle\sum_{j \text{ neighbors of } i} k_{ij}/d_{ij} \\
&\quad b(i) = \textstyle\sum_{j \text{ neighbors of } i \text{ and } j \in \Lambda} k_{ij}/d_{ij} \cdot h_0 \\
&\text{if } i \in \Lambda \\
&\quad A(i,i) = 1 \\
&\quad b(i) = h_0
\end{aligned}
\tag{3}
$$

with k_{ij} and d_{ij} the permeability and the length of the fracture between the two connected nodes i and j and h_0 is the fixed head on the limit Λ. With this formulation, A is symmetric positive definite. Iterative conjugate-gradient methods were tested and compared to the LU factorization. Iterative conjugate-gradient methods were implemented within PETSc (see footnote 1). They were either not preconditioned or preconditioned with either Jacobi's method or the incomplete LUs method. The direct decomposition method was implemented within UMFPACK (see footnote 2), a multifrontal decomposition method. Multifrontal methods are right-looking methods where the pivots can be selected both on the basis of sparsity and numerical accuracy. The factorization is performed in a sequence of frontal matrices which are small dense submatrices (Davis and Duff, 1999).

For all 2D fracture networks tested, the direct method was several orders of magnitude faster than the iterative methods whatever the preconditioning. For the network of Fig. 1A, the direct method is more than 3.5 orders of magnitude faster than the conjugate gradient preconditioned by an incomplete LU decomposition. Direct methods require an additional storage which is close to the space used to store the matrix A, whatever the network type. For a network made up of 700 000 points, the iterative methods solves the system in 36 h whereas the multifrontal method takes 73 s, i.e. around 2000 times less.

Direct methods are classically faster than iterative methods but require more memory. In the example studied here, the additional memory storage is acceptable even for large networks. However the large difference of performance between iterative and direct methods was not expected. It comes from the broad spectrum of matrix A. The matrix spectrum is the range covered by the eigenvalues of the matrix. It conditions the performance of the iterative methods. The largest is the spectrum, the slowest are iterative methods. For the network shown in Fig. 1, the spectrum of A spans several orders of magnitude (Fig. 3). Quantitatively, the width of the spectrum can be estimated by the condition number of A as given by the Matlab function "condest". It is equal to 10^8 when all fractures have the same permeability and to 10^{13} when fracture permeabilities are lognormally distributed with a logarithm standard deviation $\sigma(\log K)$ equal to 1. These large values of the condition number of A are found for all fracture networks, whatever their number of fractures, fracture length and permeability distributions. They can be explained by the direct calculation of the eigenvalues corresponding to the nodes located at the end of the fractures. As on the line corresponding to one of these points there is only one non-zero off-diagonal element, the eigenvalue is equal to the diagonal element k_{ii}/d_{ii} (Eq. (3)). The broad range of distances between nodes d_{ii} and the large variability of the fracture permeabilities k_{ii} explain the large spectrum of eigenvalues.

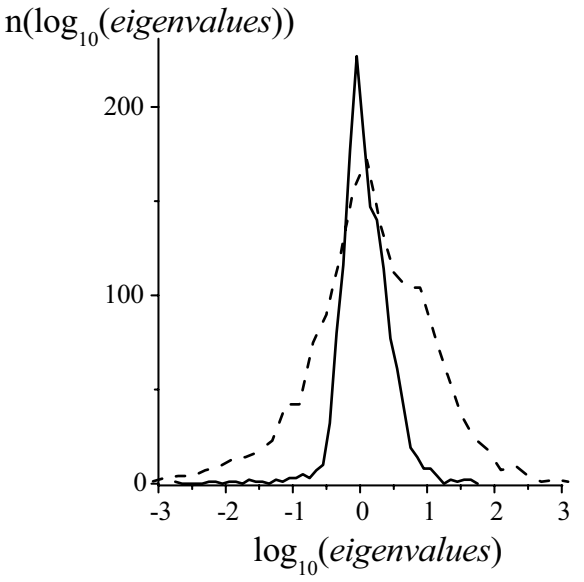

Fig. 3. Histogram of eigenvalues of matrix A for network shown in Fig. 1A with a constant fracture permeability (solid line) and with a lognormal fracture permeability distribution (dashed line).

As fracture length and aperture variabilities are typical characteristics of fractured media whatever the dimension of the network, we expect direct algorithms to be faster than iterative algorithms in 3D as well as in 2D. However, the number of neighbors per fractures could increase so much in 3D networks that the space requirements of the direct methods might be prohibitive. We are currently studying the steady-state flow problem for 3D networks.

References

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

Bour, O., Davy, P., 1997. Connectivity of random fault networks following a power law fault length distribution. Water Resources Research 33 (7), 1567–1583.

Clauser, C., 1992. Permeability of crystalline rock. EOS Transactions of American Geophysical Union 73 (21), 237–238.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., 1990. Introduction to Algorithms. MIT Press, Cambridge, MA, 1019pp.

Davis, T.A., Duff, I.S., 1999. A combined unifrontal/multifrontal method for unsymmetric sparse matrices. ACM Transactions on Mathematical Software 25 (1), 1–19.

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

de Marsily, G., 1986. Quantitative Hydrogeology: Groundwater Hydrology for Engineers. Academic Press, Orlando, FL, 464pp.

Gelhar, L.W., Welty, C., Rhefeldt, K.R., 1992. A critical review of data on field-scale dispersion in aquifiers. Water Resources Research 28 (7), 1955–1974.

Gondran, M., Minioux, M., 1984. Graphs and Algorithms. Wiley, New York, NY, 670pp.

Preparata, F.P., Shamos, M.I., 1985. Computational Geometry: An Introduction. Springer, New York, NY, 376pp.

Snow, D.T., 1969. Anisotropic permeability of fractured media. Water Resources Research (5), 1273–1289.

Stauffer, D., Aharony, A., 1992. Introduction to Percolation Theory, 2nd Edition. Taylor & Francis, Bristol, UK, 181pp.