An Efficient Parallel Mixed Method for Flow Simulations in Heterogeneous Geological Media

Hussein Mustapha * Abir Ghorayeb † Kassem Mustapha ‡ Pierre Saramito §

Abstract

The permeability of a 3D geological fracture network is determined by triangulating the fractures and solving the 2D Darcy's equation in each fracture. Here, the numerical modelling aims to simulate a great number of networks made up of a great number of fractures i.e. from 10^3 to 10^6 fractures. Parallel computing allows us to solve very large linear systems improving the realism of simulations. Several algorithms to simulating fluid flow are proposed for the cases of significant matrix permeability. In the case of a weak permeability matrix, the flow is focused in the fractures having a strong permeability and fluids percolate through networks of interconnected fractures. In this paper, we present a complete parallel algorithm for solving flow equations in fracture networks. We consider an imprevious matrix. The different parts of the algorithm are detailed. Numerical examples using the mixed finite element (MFE) method for various fracture networks illustrate the efficiency and robustness of the proposed algorithm. To the best of our knowledge, results for parellel simulation of fluid flow in discrete-fractured media with impervious matrix using the MFE method are the first to appear in the literature.

Keywords: parallel computing, geological media, fluid flow, triangular mesh, mixed finite element method.

1 Introduction

The prediction of natural underground flow circulation has brought up the concern of medium heterogeneity. Solid rock masses are in general fractured and fluids can percolate through networks of interconnected fractures [24] which are also heterogeneous and multi-scale (Fig. 1). The fractures can sometimes appear

*Corresponding author, McGill University - Montreal - Canada, and Reservoir Engineering Reseach Institute - Palo Alto- USA. Hussein.Mustapha@mcgill.ca

†TIMC Laboratory University - Grenoble - France.

‡Department of Mathematical Sciences, King Fahd University of Petroleum and Minerals - Dhahran - Suadi Arabia.

§CNRS, LJK-IMAG, Grenoble, France.

insulated, but are in general grouped in a network whose geometry depends on the principal directions of the deformation [1]. In addition, the very great variability in length of the fractures, from one millimeter to the hundreds of kilometers increases the complexity of the fractured media. A general overview of stochastic generation of fractured media problems is given in [1, 14].

Our future goal is to extend the computation for more complex mathematical model. Currently, several simulators use the dual-porosity/dual-permeability models ([3, 8, 21, 22]) for naturally fractured media, in which matrix blocks are surrounded by regular fracture patterns. Despite the numerical efficiency of this approach, it has many limitations. The accuracy depends on the description of the exchange functions between the matrix and the fractures. These functions may not be properly defined with gravity, compressibility, and capillary effects. The other limitation is that this approach assumes the medium to have densely connected fractures. Thus it may not be useful for describing discrete fractures. An important aspect in gas-oil gravity drainage of fractured reservoirs is the process of re-infiltration ([16]). When drained oil from an upper matrix block enters into a matrix block underneath, the process is called reinfiltration. The dual-porosity/dual-permeability models may not describe the reinfiltration. In this work, we adopt the discrete fracture model (DFM). See for more details [32, 34].

In the case of weak permeability matrix, the fluid flow is focused in the highly heterogeneous fractures, and governed by Darcy's law [23]. Several algorithms to study the fluid flow are proposed by [40, 41, 42, 4, 35]. In these papers, the authors take into account the phase matrix. In our applications, the matrix is of weak permeability and then it is not taken into account as in [5, 6, 7, 11, 13, 23, 24, 39] to speed up the computation. Then the resulting geometry is composed by a network of fractures disposed in 3D space. Naturally, each of these fractures (Fig. 1) represents a sub-domain, and the fluid flows in the network across the intersections between fractures. Thus, the flow in the network can be deduced from the flow in the fractures and their intersections [32, 25, 39].

For parallel simulations, the fractures can be distributed and treated separately on the processors. Each of these fracture should be treated without decomposition. Then, because the fractures exist with varying sizes [14, 6, 7, 11, 13], the principal problem is related to the load balancing on the processors. In this work, we choose to distribute the fractures on the processors with respect to their sizes (Algorithm 1) as we detail in section 3.1. After, the triangulation of each package of fractures is generated following Algorithm 2 of section 3.2. This algorithm is used in [32, 27, 30, 31]. We note that, the triangular mesh of the 3D fracture network is rather complex Fig. 3, with interconnected two-dimensional (2D) triangular meshes in each fracture (Fig. 2).

The numerical simulations consist of three main phases: generation of the linear system, solution of the linear system, and evaluation of flow. The first phase is obtained by discretizing the governing

equations which are the mass conservation equation and Darcy's law for steady-state, incompressible and single-phase flow in porous media. In this work, we use a robust and efficient numerical model that has the following features: 1) the mass is conserved locally at the element level, 2) the velocity field is correctly approximated in anisotropic and in highly heterogeneous media, and should have low mesh dependence, 3) unstructured grids are used for spatial discretization. To fulfill these requirements, we adopt the MFE method. The MFE method is used to discretize Darcy's law. The main features for this choice are: the pressure and the fluxes are approximated simultaneously with the same order of convergence; the method is locally conservative and it can easily accommodate full permeability tensor. This method also produces minimal mesh orientation effect [12]. The fact that the MFE method is more reliable in flux calculation than the finite volume (FV) and the finite element (FE) methods is well known [15, 26]. The MFE formulation in our method uses the lowest order Raviart-Thomas space [36, 38]. The original MFE formulation leads to a saddle-point problem for elliptic or parabolic equations i.e., the linear system to solve, that has the cell-pressure averages and the inter-element fluxes as primary unknowns, is indefinite. The remedy is to use the hybridization technique [9, 10] where new degrees of freedom are appended at the element edges. The additional unknowns represent the edge pressure averages (pressure traces). For more details, see [32]. Numerically, the discrete problem to solve is linear, with a sparse symmetric positive definite matrix. The very strong variability of hydraulic properties leads to an ill-conditioned matrix. The numerical study of the influence of this kind of heterogeneity on the underground flow circulation needs to generate a large number of realistic numerical simulations leading to very large sparse linear systems. In order to reach this objective, we have to overcome two main problems: the memory size to generate very large linear systems, and the CPU time to solve a large number of linear systems. The research of scaling laws [23, 14] in the fractured mediums requires the calculation of flow in a great number of networks containing a great number of fractures, which strongly influences the size of the linear systems. Consequently, the CPU time and the memory capacity become significant for the simulation of full-scale problems.

Here, parallel computing is used as an alternative, making it possible to obtain substantial reductions in the CPU time and to carry out numerical studies of a higher degree of accuracy while keeping a reasonable CPU time. The possibilities of parallel computing are today present to differing degrees in the information processing systems. High performance computing is thus mandatory in this framework.

This paper is organized as follows: In section 2, we present the mathematical model and our motivation for using parallel computing. In the third section, we describe the details of each part of the developed algorithm. Some numerical results are presented in section 4. Finally, a summary and a preview of future

work are given in section 5.

2 Mathematical model

Each fracture (originally a threedimensional object) is approximated by a flat ellipse characterized by its middle coordinates, orientation, hydraulic permeability or aperture distribution, and wall roughness. However, due to the high computer requirements, it is only possible to solve local problems by using stochastic discrete fracture network models. We refer to [39] for more details. The network is included in a cube of size L, the fractures are modelled by ellipses. The fractures length is modelled by a random power-law distribution. We define the fracture network R by $R = \bigcup_{i=1,NF} \bar{F}i \backslash \partial R$. R is the bounded domain with boundary $\partial R = \Gamma^D \cup \Gamma^N$, and Γ^D (resp. Γ^N) corresponds to the Dirichlet (resp. Neumann) boundary conditions. Fi is the fracture i, NF is the total number of fractures. We seek the fracture flow velocity u (a two-dimensional vector in each Fi), which is the solution of the problem:

$$u = -K(\nabla \overline{h} + z) \quad \text{in } R, \tag{2.1}$$

$$\nabla . u = f \quad \text{in } R, \tag{2.2}$$

$$\overline{h} = \overline{h}^D \quad \text{in } \Gamma^D, \tag{2.3}$$

$$u.n = u_N \quad \text{in } \Gamma^N, \tag{2.4}$$

where all the variables are expressed in the local coordinates of appropriate Fi, and the differentiation is always done with respect to these local coordinates. Eq.(2.1) is Darcy's law, Eq.(2.2) is the mass balance equation, and Eqs.(2.3)-(2.4) describe Dirichlet and Neumann boundary conditions. The variable $\overline{h}$ denotes the piezometric head, $\overline{h} = \overline{p}/\rho g$, where $\overline{p}$ is the fluid pressure, g is the gravitational acceleration constant, ρ is the fluid density, f represents stationary sources or sinks density, and z is the elevation, i.e. the upward vertical three-dimensional coordinate. The second-rank tensor K of hydraulic conductivity is a function of the original three-dimensional fracture aperture, wall roughness, and filling. We suppose that K is symmetric and uniformly positive definite on Fi.

3 Parallel algorithm description

The developed algorithm contains four essential parts. In the first part, we generate the fractures and we distribute them on the processors. In the second part, we generate the mesh for the package of fractures

on the processors. Using the MFE method, we discretize the flow equations and we construct the local linear system associated at each processor, in the third part. Finally, we solve the global linear system that is distributed on the processors and we compute the flux. Following, we explain each of these steps in detail.

3.1 Distribution of fractures into processors

The criterion of distribution used in this work is related to the accumulated surface of the fractures on each processor. This distribution of fractures provides a package of fractures on each processor. The efficiency of a parallel algorithm strongly depends on an equitable distribution of the load of calculation between the processors. To explain the adopted algorithm with a simple way, we give the following example. Denote by p the number of the processor, $0 \leq p \leq P-1$, where P is the total number of processors. Consider a network of 10 fractures (i.e., from F1 to F10). We assume that the sizes of the fractures are F1=1 uos (i.e., uos is a unit of surface), F2=10 uos, F3=5 uos, F4=110 uos, F5=300 uos, F6=50 uos, F7=120 uos, F8=240 uos, F9=400 uos and F10=7 uos, respectively. If $P > 10$, then the fractures can be distributed on the first 10 processors. If $P < 10$, then we make the following steps. In the first step, we sort the fractures with respect to their sizes. We obtain the sequence (F9=400 uos, F5=300 uos, F8=240 uos, F7=120 uos, F4=110 uos, F6=50 uos, F2=10 uos, F10=7 uos, F3=5 uos, F1=1 uos). Assume that, for example, $P = 3$. In the second step, we distribute the first three (i.e., in general the first P) fractures (i.e., F9=400 uos, F5=300 uos and F8=240 uos) on the processors ($p0, p1$ and $p2$), respectively. In the third step, we distribute the remaining fractures. In this step, for each of the remaining fractures we seek the target processor that has the minimal charge (e.g., the accumulated surface of the fractures, pertaining to the processor, is smallest) to allot a new element. For example: F7=120 uos is sent to $p2$ (F8=240 uos and F7=120 uos are on $p2$, then we have a total of 360 uos), F4=110 uos is sent to $p1$ (F5=300 uos and F4=110 uos are on $p1$, then we have a total of 410 uos), F6=50 uos is sent to $p0$ (F9=400 uos and F6=50 uos are on $p0$, then we obtain a total of 450 uos). After, the fractures F2=10uos, F10=7uos, F3=5uos and F1=1 uos are sent to $p1$ because it has the minimal value. Finally, we obtain 450 uos (i.e., the package F9 and F6) on $p0$, 410 uos (i.e., the package F5 and F4) on $p1$ and 383 uos (i.e., the package F8, F7, F2, F10, F3 and F1) on $p2$. The general algorithm can be presented as follows:

Algorithm 1: Distribution of fractures

We assume that $NF > P$.

1. *Sort the values of fractures area;*

2. *Distribute the first P fractures on the processors: the fracture Fi is sent to the processor i;*

3. Send the fracture Fi, for $P < i \leq NF$, *to the processor that has a minimal charge (e.g., the accumulated surface size of fractures on that processor is the smallest by comparing to the accumulated surface size on each of the other processors). This step is repeated until i reaches NF.*

Algorithm 1 can be applied in all the problems containing heterogeneous objects. Different algorithms are developed in the literature for this type of problems. For example, in [20], the authors developed a method based on the partition of unstructured graph and took into account the communication between the elements (i.e., the fractures). This method is very efficient when the size ratio between fractures is not very big. But as we mentioned before, the length of the fractures varies from one millimeter to the hundreds of kilometers. Using Algorithm 1, the fractures are distributed with a best way in term of load balancing. Then, the gain in CPU time in triangular mesh and in the construction of the linear system compensates the lost in CPU time in the communications between processors. For a general application, Algorithm 1 can be improved by compromising between the load balancing and the communications between processors.

3.2 Triangulation of fractures

A 2D mesh of each fracture is then generated by using AF2FM software [33]. It ensures that the intersections are equally discretized in common fractures (Fig. 2). A similar method is developed in [36], but with non-matching discretized intersections and an adapted nonconforming MFE method. The equations are then approximated by using the mesh and leading to a linear system of equations. In the fracture networks we have small and large fracture sizes. In this paper, we have used what we called the conforming MFE method. Then, the intersections between fractures have the same discretization in the fractures containing them. For example, an intersection between two fractures is decomposed into the same segments in the two fractures. Then the fractures are for the moment discretized using the same mesh step h. This mesh step was chosen by respecting the size of the fractures and the connections between them. This step is independent from the original topic discussed here. Then the fracture's size is the principal key, and our algorithms are related strongly to the fracture's area.

In order to get a symmetric positive definite matrix, a hybrid approach is used [19]. The order of the matrix is the number of edges in the network mesh. The parallel mesh generation relies on a data distribution of fracture structures. In order to get a static balanced task scheduling, we implement a variant of the Bin packing algorithm [2]. The mesh generation is completely parallel; the communications occur only to attribute global numbers to mesh edges. Then we infer the data distribution of the mesh and

the matrix structures from the parallel mesh generation. Therefore matrix generation is done in parallel with the same distribution. All fracture intersections are processed by one unique processor, which collects matrix data related to intersections from other processors. Denote by $Frac(p)$ the package of fractures on the processor p. Each processor has locally a data structure, which contains the triangulation of $Frac(p)$. The triangular mesh of the network is obtained by meshing the packages $Frac(p)$, $p = 0 \ldots, P-1$. After the triangulation, we obtained local numbers of edges related to each fracture. To build the global linear system, we allot global numbers to these edges. Each edge intersection has a global number already allotted.

The various steps of the algorithm carrying out this part of the code are detailed below:

Algorithm 2: Triangulation of fractures and edges global numbering

ng : is a vector allocated on the processor 0 and it is of size P;

$nF(p)$: is the number of fractures on the processor p;

$na^{(i,p)}$: total number of boundary and internal edges of Fi on the processor p;

$na\_tot^{(p)}$: total number of boundary and internal edges on the processor p.

1. **For** each processor p :

   (a) **If** $p = 0$

      i. **For :** $i = 1, ..., nF(0)$

         A. *triangulation of* $Fi \in Frac(0)$

         B. $na^{(i,0)}$ = *total number of boundary and internal edges of* Fi

         C. $na\_tot^{(0)} = na\_tot^{(0)} + na^{(i,0)}$

         End For

      ii. ni = *total number of intersection edges in the network*

      iii. $ng(0) = ni + 1$

      iv. **For :** $p = 1, ..., P-1$

         A. *receive* $na\_tot^{(p)}$ *from processor* p $(p = 1, ..., P-1)$

         B. $ng(p) = ng(p-1) + na\_tot^{(p)}$

         C. *send* $ng(p)$ *to processor* p

         End For

   Else

      i. **For** $i = 1, ..., nF(p)$

A. *triangulation of* $Fi \in Frac(p)$

B. $na^{(i,p)}$= *total number of boundary and internal edges of* Fi

C. $na\_tot^{(p)} = na\_tot^{(p)} + na^{(i,p)}$

End For

ii. *send* $na\_tot^{(p)}$ *to processor 0*

iii. *receive* $ng(p)$ *from processor 0*

End If

End For

2. **For** each processor p :

(a) *Global numbering of fractures' edges starting from* $ng(p)$.

End For

3.3 Local linear system building

The construction of the linear systems results from the distribution of the global linear system lines associated at the fracture network. The matrix is distributed by rows on the processors. Each line corresponds to an edge in the mesh. We use the distribution of the edges induced by the distribution of the fractures and the mesh. We distinguish the intersections edges which are common to several fractures and the internal edges which are local with the fractures. On each processor, we build the partitioned matrix and second member

$$A(p) = \begin{bmatrix} A_0(p) \\ A_1(p) \end{bmatrix} \text{ and } b(p) = \begin{bmatrix} b_0(p) \\ b_1(p) \end{bmatrix}$$

where $A_0(p)$ gathers all the edges intersections in $Frac(p)$, and $A_1(p)$ gathers all the internal and boundary edges in $Frac(p)$. We note by $A^{(i,p)}$ the corresponding block at all internal edges in the fracture i of $Frac(p)$. The global linear system is of the following form

$$\begin{bmatrix} A_0 \\ A_1(0) \\ A_1(1) \\ . \\ . \\ A_1(P-1) \end{bmatrix} * \begin{bmatrix} X_0 \\ X_1(0) \\ X_1(1) \\ . \\ . \\ X_1(P-1) \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1(0) \\ b_1(1) \\ . \\ . \\ b_1(P-1) \end{bmatrix}$$

where X_0 contains all the unknown factors of the intersections between fractures. The matrix $A_1(p) = \sum_{Fi \in Frac(p)} A^{(i,p)}$ $(p = 0 \ldots, P-1)$ is allocated on the processor p. We choose to gather all the unknown factors, related to the edges intersections, in A_0 and b_0 and to allocate them on processor 0. To store the linear system, we chose storage by coordinates [37]. The build of A_0 requires communication. The continuity of flow across the intersections edges is ensured by the communications between the packages of fractures through their intersections (Appendix A). Each processor sends its contribution $A_0(p)$ to the processor 0, which calculates A_0, by juxtaposing all the extra-diagonal non zeros elements and by summoning the diagonal terms [27]. The mathematical aspects of the problem are discussed in details in [27, 28, 29, 30, 31]. The algorithm that describes this part is presented as follows:

Algorithm 3: Linear system building

1. For $i \in Frac(p)$
   (a) Compute $A^{(i,p)}$ et $b^{(i,p)}$
   End For
2. Assemble $A(p)$ and $b(p)$
3. If $p \neq 0$ then
   (a) send $A_0(p)$ and $b_0(p)$ to the processor 0
   Else
   (a) For $p = 1, \ldots, P-1$
      i. receive $A_0(p)$ and $b_0(p)$ from the processor p
      ii. integrate $A_0(p)$ and $b_0(p)$ in A_0 and b_0
   End For
   End If

We note that the number of triangles and edges is related to the accumulated surface in each processor, and then Algorithm 1 has the principle role.

3.4 Linear system resolution and flow computation

The linear system obtained from the discretization of equations with the MFE method is characterized by a sparse symmetric positive definite matrix. Here we mention that the algorithms 1, 2, and 3 are independent from the linear solver. Then these algorithms are not related to the use of a specific type of linear solvers. In this paper, we consider the use of a direct or an iterative solver out of our main discussion. However, the data can be arranged to correspond to any external linear solver. As we mentioned before, the complexities in geometry and hydraulic properties leads to an ill-conditioned matrix. In this paper, we have used a direct linear solver. The direct method is based on the Cholesky factorization $A = UU^T$, which is accurate and robust. We use the Pspases library [17], which is a very powerful parallel sparse direct solver devoted to symmetric positive definite matrices. Parallelism is based on a distributed-memory paradigm and communications are handled by the MPI library [18]. A slight drawback is that the number of processors must be a power of 2, and that there is no sequential version of Pspases. Once the linear system has been solved, we compute the hydraulic head and the flux on each element of the computational mesh. As for the hydraulic conductivity, the hydraulic head is distributed on the processors. Each processor has the value of hydraulic head on its computational sub-domain.

4 Numerical results

All tests are performed using a SUN cluster composed of two nodes of 64 computers each. Each computer is a 2.2 Ghz AMD Opteron bi-processor with 2 G-byte of RAM. Inside each node, computers are interconnected by a Gigabit Ethernet Network Interface, and the two nodes are interconnected by a Gigabit Ethernet switch (CISCO 3750).

Denote by T_P the parallel run time (in seconds) on P processors. In the tests we used three fracture networks with, respectively, 1000, 12000 and 25000 fractures. As we explained before, the distribution of fractures is the main part of the algorithm. Then, in the first test, we evaluate the CPU using a public algorithm developed in [20]. The results are reported in Table 1. This results show that T_2/T_{32} varies form 12 to 15, for 1 000, 12 000 and 25 000 fractures, in the time mesh generation. Then, the efficiency here, E_{16}, is about 0.95. In the other side, T_2/T_{32} varies between 11 and 13 in the construction of the linear system. Then a good efficiency $E_{16} \simeq 0.9$ is obtained here. Now, using Algorithm 1 to distribute the fractures, we obtained the results in Table 2. These results are good and close to the above results. Here, we mention that T_P is about two times less using Algorithm 1 than the public algorithm. In fact, the construction of the linear system is in the next step after the mesh generation. Then, to start this

step we should wait all the processors to finish the first step (the mesh generation). In this case, T_P is evaluated like the maximum CPU time on the P processors. For the mesh generation, T_P is related more to the fractures' surfaces. Then more the accumulated surfaces of fractures on the processors are close, better is the parallel run time. As example, we present in Fig.4, the head piezometric profile.

The number of fractures can be more than the numbers that used in the paper. The steady state flow is a simple problem test. This algorithm will be extended to solve transient, water injection, CO2 injection, compositional problem, etc. In the first side, we want to simulate many cases for different type of fracture networks and different number of fractures for scaling law. In the other side and for more complex mathematical models, we need to run the code for many time steps.

5 Conclusion

The numerical simulation is an essential tool for the technological development of very varied disciplines like the mechanics of fluids, the structural analysis, and geology. A parallel algorithm to compute flow fluid in 3D discrete fracture network is presented. We showed that the parallelism is essential to treat networks of big size. The study of this type of networks is necessary for the research of scaling laws. We compared the parallel run time with a public algorithm. In the future work, we envisage to optimize the algorithms and especially Algorithm 1. The aim is to obtain an optimal algorithm which compromises between the size of distributed objects and communications.

6 Appendix A

Discretization of Darcy's law. The essential idea of the mixed methods is to approximate individually the Darcy's law and flow equation and we get additionally the Darcy velocity as an unknown function. Thus, the variation formulations of the given PDEs systems are chosen in a way to have the pressure and its gradient in the basic formulation. The Raviart-Thomas space of lowest order ($RT0$) is used to approximate the velocity [32, 36]. The main idea is to express the velocity over each grid cell with respect to the fluxes across the cell edges. Based on Raviart-Thomas approximation space, the vectors u in Eq.(2.1) can be expressed as:

$$u = \sum_{b} \sum_{E \in \partial b} q_{b,E} W_{b,E} \tag{A.1}$$

where $W_{b,E}$ is a $RT0$ basis function, $q_{_{b,E}}$ is the total flux across an edge E. These vectors are determined by their normal fluxes across the cell edges. By inverting K, Darcy's velocity equation becomes

$$K^{-1}u = -(\nabla p + \nabla z) \tag{A.2}$$

We denote by E' (resp. b' or b'_i) an edge (resp. a triangle) of the triangulation mesh. Multiplying Eq.(A.2) by the test function and integrating by parts, the total flux is suppressed through each edge E as a function of the cell pressure-average p_b and the edge-pressure averages $tp_{b,E}$ for each triangle b, i.e.,

$$q_{b,E} = \alpha_{b,E} p_b - \sum_{E' \in \partial b} (B_b^{-1})_{E,E'} tp_{b,E'}, E \in \partial b \tag{A.3}$$

where B_b is a $3 * 3$ symmetric positive definite matrix whose elements are

$$B_b = \int_b W_{b,E}^T K^{-1} W_{b,E'} \text{ and } \alpha_{b,E} = \sum_{E' \in \partial b} (B_b^{-1})_{E,E'} \tag{A.4}$$

The continuity of the fluxes across the inter-element boundaries provides:

$$q_{b,E} = \left\{ \begin{array}{ll} \displaystyle\sum_{E',E' \in \partial b'} q_{_{b',E}} & \text{if } E \notin \Gamma^N, E \in \displaystyle\bigcap_{i=1,nbE} b'_i \\ q^{N,E} & \text{if } E \notin \Gamma^N \end{array} \right| \tag{A.5}$$

where nbE is the number of triangles which contain the edge E. We note that $nbE > 2$ if E is an intersection edge between fractures. All the complexities come from the realization of the fluxes continuity in Eq.(A.5). For example, we consider an intersection edge E (between two fractures or more) which is on the processor p. Because the fractures are distributed on the processors then it is hard (in terms of CPU time and structures analysis) to search the nbE triangles that contain E in all the processors and to send them to the processor p to be able to provide the condition in Eq.(A.5).

References

[1] Adler, P.M. and J.-F. Thovert, 1999 Fractures and Fracture Networks (Kluwer Academic, Dordrecht, 1999).

[2] Albers, S. and M. Mitzenmacher, 2000. Average-case analyses of first fit and random fit bin packing. Random structures alg. 16, 240-253, 2000.

[3] Arbogast, T., J. Douglas, and U. Hornung (1990), Derivation of the double porosity model of single phase via homogenization theory, SIAM J. Math. Anal., 21, 823-836.

[4] Bastian, P., C. Zhangxin, E. Richard, H. Rainer, J. Hartmut and V. Reichenberger, 2000. Numerical Simulation of Multiphase Flow in Fractured Porous Media. Lecture Notes in Physics, 552, 50.

[5] Billaux, D., J.P. Chiles, K. Hestir and J. Long, 1989. Three-dimensional statistical modelling of a fractured rock mass – an example from the Fanay-Augères mine, Int. J. Rock Mech. Min. Sci. and Geomech. Abstr, 26, 281.

[6] Bonnet, E., O. Bour, N. Odling, P. Davy, I. Main, P. Cowie, and B. Berkowitz, 2001. Scaling of Fracture Systems in Geological Media, Reviews of Geophysics, 39 (3), 347-383.

[7] Bour, O. and P. Davy, 1998. On the connectivity of three dimensional fault networks. Water Resources Research, 34(10), 2611-2622.

[8] Bourgueat A. (1984), Homogenized behavior of diphasic flow in naturally fissured reservoir with uniform fractures, Comp. Methods in Applied Mechanics and Engineering, 47, 205.

[9] Brezzi, F., and M. Fortinn, 1991. Mixed and Hybrid Finite Element Method, Springer-Verlag, New York.

[10] Chavent, G., and J.-E. Roberts, 1991. A unified physical presentation of mixed, mixed-hybrid finite element method and standard finite difference approximations for the determination of velocities in water flow problems, Adv. Water Resour., 14(6), 3.

[11] Cacas, M.C., and al., 1990 Modeling fracture flow with a stochastic discrete fracture network: calibration and validation. 1. The flow model, Water Resources Research, 26, 479.

[12] Darlow, B., R. Ewing, and M. Wheeler, 1984. Mixed finite element method for miscible displacement problems in porous media. SPEJ, 24, 391–398.

[13] De Dreuzy, J.R., P. Davy, and O. Bour, 2000. Percolation threshold of 3D random ellipses with widely-scattered distributions of eccentricity and size, Phys. Rev. E, 62 (5), 5948-5952.

[14] Dietrich, P., R. Helmig, M. Sauter, H. Hötzl, J. Köngeter, and G. Teutsch, 2005. Flow and Transport in Fractured Porous Media, 2005 XVIII, 447 p. 3-540-23270-2. Berlin: Springer.

[15] Durlofsky, L, 1994. Accuracy of mixed and control volume finite element approximations to darcy velocity and related quantities. Water Resour. Res., 30(4), 965.

[16] Firoozabadi, A., K. Ishimoto, and B. Dindoruk (1994), Reinfiltration in fractured porous media: Part 2 - Two Dimensional Model, SPE Advanced Technology Series, 2, 45-51.

[17] Gupta, A., F. Gustavson, M. Joshi, G. Karypis, and V. Kumar, 1999. Pspases: An efficient and scalable parallel sparse direct solver. In parallel numerical computations with applications, T. Yang (ed.), Kluwer international series in engineering and computer science, 515.

[18] Hansen, {P. B.,1998. An evaluation of the message-passing interface, ACM Sigplan Notices, 33 (3), 65–72.

[19] Hoteit, H., J. Erhel, R. Mosé, B. Philippe and P. Ackerer, 2002. Numerical reliability for mixed methods applied to flow problems in porous media. Computational geosciences 6(2), 161-194.

[20] Karypis, G. and V. Kumar, METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices Version 4.0. University of Minnesota, Department of Computer Science, Army HPC Research Center, Minneapolis, MN 55455, 1998.

[21] Kazemi, H., and J. Gilman (1969), Pressure transient analysis of naturally fractured reservoirs with uniform fracture distribution, SPE J., 9, 451–462.

[22] Kazemi, H., and J. Gilman (1992), Analytical and numerical solution of oil recovery from fractured reservoirs with empirical transfer functions, SPE J., 219-227.

[23] Koudina, N., R. Gonzalez Garcia, J.-F. Thovert and P.M. Adler, 1998. Permeability of three-dimensional fracture networks. Phys. Rev. 57, 4466-4479.

[24] Neretnieks, I., 1985. Transport in fractured rocks. Proceedings of Hydrogeology of Rock of Low Permeability, Mem. Intern. Assoc. Hydrogeol., 17(2): 301-318.

[25] Maryška, J., O. Severýn and M. Vohralík, 2004. Numerical simulation of fracture flow with a mixed-hybrid FEM stochastic discrete fracture network model, Computational Geosciences, 8, 217.

[26] Mosé, R., P. Siegel, P. Ackerer, and G. Chavent, 1994. Application of the mixed-hybrid finite element approximation in a ground water flow model: Luxury or necessity?. Water Resour.Res., 30(11), 3001.

[27] Mustapha, H., 2005a. Simulation numérique de l'écoulement dans des milieux fracturés tridimensionnels. Thèse de Doctorat, Université de Rennes 1. http://www.irisa.fr/centredoc/publis/theses#2005.

[28] Mustapha, H., A. Beaudoin, J. Erhel and J.R. de Dreuzy, 2005b. Parallel Simulations of Underground Flow in Porous and Fractured Media, International conference in computing parallel, PARCO, Malaga, Spain.

[29] Mustapha, H., J. Erhel and J.R. de Dreuzy, 2005c. Heterogeneous Fractured Media: Mathematical Analysis and Parallel Computing, International Conference on Parallel and Distributed Processing Techniques and Applications, Nevada, USA.

[30] Mustapha, H., 2006a. An Adaptive method for Flow Simulation in Three-Dimensional Heterogeneous Discrete Fracture Networks", in the International Conference on Scientific Computing, Nevada, USA.

[31] Mustapha, H., 2006b. Monte Carlo Simulation of Flow in Three-Dimensional Discrete Fracture Networks, in the 2006 World Congress in Computer Science, Computer Engineering, and Applied Computing, Las Vegas, USA.

[32] Mustapha, H. and K. Mustapha, 2007. A New Approach to Simulating Flow in Discrete Fracture Networks with an Optimised Mesh, SIAM J. SCI. COMPUT. , 29 (4), 1439–1459.

[33] Mustapha, H., 2007. AM2FM : Autmotical Generator Mesh for Complex 2D Discrete Fractured Media, user's manual 1.0.1.

[34] Mustapha, H., and A. Firoozabadi, 2007. Simulation of Black-Oil Two-phase Flow in Fractured Media Without Transfer Functions (submitted).

[35] Philip, Z. G., J. W. Jr. Jennings, J.E. Olson, S.E. Laubach and J. Holder, 005. Modeling coupled fracture-matrix fluid flow in geomechanically simulated fracture networks. SPE Reservoir Evaluation & Engineering, 8, 4, 300-309.

[36] Raviart, P.A. and J. M. Thomas, 1996. A mixed hybrid finite element method for the second order elliptic problem. in Lectures Notes in Mathematics 606, 292-315.

[37] Saad, Y., Iterative Methods for Sparse Linear Systems. PWS Publishing Company.

[38] Thomas, J., 1977. Sur l'Analyse Numérique des Méthodes d'Elément Finis Hybrides et Mixtes. Thèse de Doctorat d'Etat, Université de Pierre et Marie Curie.

[39] Vohralik, M., 2004. Méthodes numériques pour des équations elliptiques et paraboliques non linéaires: Application à des problèmes d'écoulement en milieux poreux et fracturés. Thèse de Doctorat, Université Paris XI Orsay.

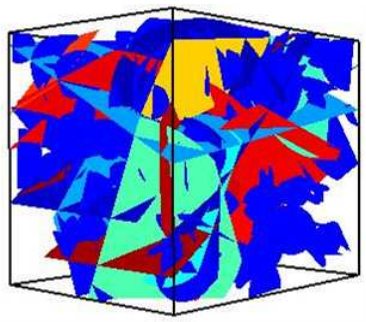

Figure 1: FN : 500 fractures generated in a cube size equal to 100 m^3.

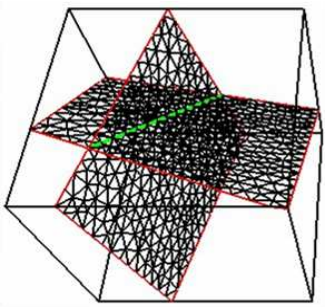

Figure 2: Triangulation mesh for two fractures in 3D with an impervious matrix

[40] Yu-Shu, W., K. Zhang, D. Chris, K. Pruess, E. Elmroth and S. Bodvarsson, 2002a. An efficient parallel-computing method for modeling nonisothermal multiphase flow and multicomponent transport in porous and fractured media. LBNL-47937, Advance in Water Resources, 25, 243-261.

[41] Yu-Shu, W., K. Zhang, and K, Pruess, 2002b. Massively Parallel Simulation of Flow and Transport in Porous in Variably Saturated Porous and Fractured Media. LBNL-49407, Developments in Water Science, 47, Edited by S. M. Hassanizadeh, R. J. Schotting, W. G. Gray, and G. F. Pinder, Computational Methods in Water Resources, 1, 289-296.

[42] Zhang, K., Yu. Shu. W. and G. S. Bodvarsson, 2003. Massively Parallel Computing Simulation of Fluid Flow in the Unsaturated Zone of Yucca Mountain, Nevada. LBNL-48883, Journal of Contaminant Hydrology, 381-399.

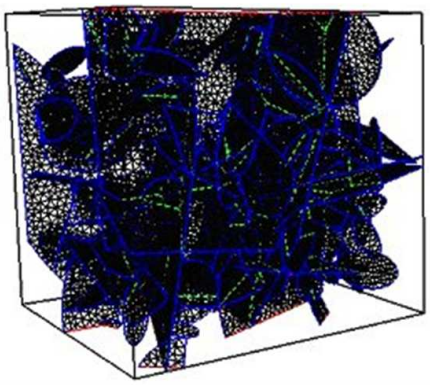

Figure 3: Example of triangulation of the network NF.

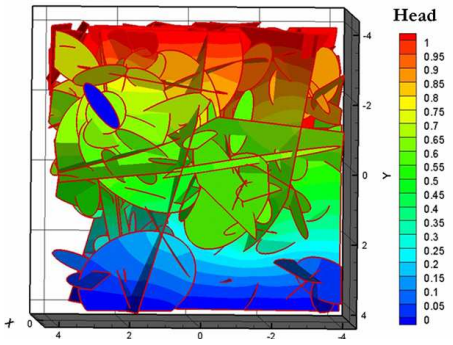

Figure 4: Example of flow computation in the network NF.

NF	N	Mesh+linear system				
		T_2	T_4	T_8	T_{16}	T_{32}
1000	0.3×10^6	9.72	5.20	2.80	1.60	0.90
12000	3.3×10^6	97.30	52.83	28.59	14.85	7.78
25000	6.2×10^6	213.79	112.77	58.5	31.12	16.4

Table 1: Results of parallelism obtained with respect the number of processors for various number of fractures. Using the algorithm in [20]. The CPU time unit is the seconds.

NF	N	Mesh + linear system				
		T_2	T_4	T_8	T_{16}	T_{32}
1000	0.3×10^6	4.86	2.59	1.40	0.82	0.346
12000	3.3×10^6	50.76	26.42	13.78	7.08	3.71
25000	6.2×10^6	106.70	55.70	1276.40	15.20	7.70

Table 2: Results of parallelism obtained with respect the number of processors for various number of fractures. Using Algorithm 1. The CPU time unit is the seconds.