FRACTURE-BFC: A COMPUTER CODE FOR THE NUMERICAL PREDICTION OF HYDRODYNAMICS IN FRACTURED-POROUS MEDIA

G.M. Grandi and J.C. Ferreri*
CNEA, Gerencia de Protección Radiológica y Seguridad
Avda. del Libertador 8250
1429 Buenos Aires, Argentina

RESUMEN

Se formulan algoritmos computacionales para la solución numérica de las ecuaciones de Navier-Stokes en medios porosos fracturados. Se considera la aproximación de Darcy a las ecuaciones de movimiento y se resuelven las ecuaciones resultantes utilizanedo una técnica de diferencias finitas ajustadas a contornos. Se discuten dos algoritmos, a fin de tener en cuenta la solución del problema algebraico asociado, a saber: directo e iterativo. Se presentan varios ejemplos indicativos de las posibilidades (y limitaciones) de los programas aplicados en relación con los algoritmos presentados.

ABSTRACT

Computational algorithms for the numerical solution of the Navier-Stokes equations in fractured-porous media are formulated. Darcy's approximation to the equations of motion is considered and the resulting equations are solved using a boundary-fitted finite-difference technique. Two algorithms are discussed so as to consider the solution of the associated algebraic problem, namely: direct and iterative. Several examples are presented showing the capabilities (and the limitations) of the implemented codes in correspondence with the presented algorithms.

---

* Member of the Carrera del Investigador Científico, CONICET, Argentina.

1. INTRODUCTION

The equations governing the flow of fluids in porous media have been discussed by many authors. Among them, Sánchez-Palencia /1/ considered the validity of the usual approximations to the Navier-Stokes equations in general terms. The consideration of cases of fractured-porous media is a more realistic approach for many rock formations and plenty of literature is now available on this subject (see /2/ as an example) because of its practical interest.

This paper arose from the authors' interest in the modeling of a high-level waste repository.

In /3/, Narashiman discussed the different approaches to the modeling of the flow in fractured-porous media and presented an integral finite-difference method. Later on, in /4/, this method was extended to consider multiple interacting continua including the coupled effects of heat and multiphase fluid flow.

In /5/, the present authors introduced some details of a code dealing with the flow in porous media and suggested its extension to consider the flow in fractured-porous media. This method, employing boundary-fitted coordinates, was an adaptation of an earlier code /6/ to this type of flow, but incorporated an additional feature: namely, the use of a direct sparse solver in order to solve the resulting system of algebraic equations. Both versions of the code included the solution of the coupled energy equation.

The present paper is a full description of the algorithms employed to solve the problem of the flow in a fractured porous-media. To the author's knowledge, they introduce different features with respect to other existing algorithms.

In what follows the governing equations are considered in the case of the flow in a porous rock with discrete fractures. The validity of Darcy's and Boussinessq's approximations to the Navier-Stokes equations is accepted for the flow both in the bulk of the rock and in the fractures. The energy equation is not solved in discrete terms, but analytical approximations are assumed to hold. The discretization of the resulting equations is shown later, along with some computational details. The relative performances of the codes are discussed and some examples of typical results are included for two-dimmensional flows.

GOVERNING EQUATIONS

The equations governing the flow are: the momentum equation in the porous media,

$$\bar{u} = -\frac{Kr}{\nu_f} \left(\nabla P + \beta \bar{g} \Delta T \right) ; \qquad (1)$$

the momentum equation for the flow in the fractures,

$$\bar{u}_f = -\frac{K_f}{\nu_f} \left(\nabla P \cdot \bar{l} \right) \bar{l} + \beta \left(\bar{g} \cdot \bar{l} \right) \bar{l} \, \Delta T \; ; \qquad (2)$$

and the equation for mass conservation,

$$\int_\tau \nabla \bar{u} \; d\tau + \int_{\partial\tau} \left(\bar{u}_f \cdot \bar{n} \right) ds = 0 \qquad (3)$$

In equations 1-3, $\bar{u}$ is the volume-averaged fluid velocity, $\bar{u}_f$ is the velocity in the discrete fractures, P is a reduced (i.e. divided by ρ_0) pressure which is the sum of the static pressure of the fluid plus the static head, β es the volumetric thermal expansion coefficient of the fluid, T is the temperature in the system, g is the gravity acceleration, ν_f is the kinematic viscosity of the fluid, $\bar{l}$ is a unit vector defining the direction of the fractures, τ and $\partial\tau$ are a control volume and its boundary, respectively. Finally, K_r and K_f are the permeabilities of the porous rock and of the fractures, respectively.

The temperature is considered as a given function in equations 1-3. This restriction is accepted as a useful simplification in the case of repository modeling if the Raleigh's number of the system is not high, as it frequently occurs. It is not a very difficult task to integrate the energy equation simultaneously with equations 1-3, and in reference /5/ the authors showed some results of this procedure. In some of the implemented codes consideration was given to variations of K_r with the position. The extension to consider variable ν_f is straightforward.

For the porous matrix, the determination of k_r is simple because it is considered as data. The corresponding value for the fractures is obtained by assuming that the flow satisfies the Poiseuille flow law (cubic law for flow-rates); this, in turn, implies that:

$$k_f = b^2 / 12 \; ,$$

where b is the aperture of the fracture. In the actual codes, the permeability for the fractures was allowed to follow this law or to being filled with a porous material of a given permeability (linear law for flow-rates).

NUMERICAL METHODS

The numerical solution of 1-3 was obtained by means of boundary-fitted finite-difference techniques. The discrete version of these equations can be found from previous works by J.F. Thompson (see reference /7/, for instance) and by one of the authors /6/. As the present approach is somewhat different from the usual ones in boundary-fitted coordinates, some alge-

braic details were provided in /5/, but only for the homogeneous porous rock. The algorithms are now extended to allow for the presence of discrete fractures and, correspondingly, some modifications to the specifications in /5/, are given.

Figure 1 shows a cell belonging to a hypothetical grid in the physical plane and illustrates the location of the variables. As can be seen, velocities located at the cell corners and cell-centered pressures are considered. Temperatures are located at cell corners. D_{ij} is a discrete representation of the flow divergence and is located at the cell center.

Also shown in figure 1 are two paths crossing at the centroid of the cell, representing one-dimensional links between pressure nodes. These paths are the discrete image of the fractures in the rock. Thus, the representation of the structure of the rock consists of large blocks of homogeneous porous continuum (or an equivalent fractured rock block) sorrounded by one-dimensional fractures. The velocities along the fractures are called u_f and v_f, respectively.

It is interesting to point out that this particular centering of the variables forces the fractures to be coincident with lines joining the centroids of the cells. Then, the boundary-fitted techniques is employed to fit "internal" boundaries as well as the external ones. As a consequence, u_f and v_f can be considered as the moduli of vectors tangent to the lines of constant values of the coordinates in the computational plane.

Considering the computational plane of reference, with independent variables U and V, equations 1-3 transform into:

$$u = - \Delta t_r \left(AUX \frac{\partial P}{\partial U} + AVX \frac{\partial P}{\partial V} \right) , \tag{4}$$

$$v = - \Delta t_r \left(AUY \frac{\partial P}{\partial U} + AVY \frac{\partial P}{\partial V} + \beta g \Delta T \right) , \tag{5}$$

$$\left\{ BUX \frac{\partial u}{\partial U} + BVX \frac{\partial u}{\partial V} + BUY \frac{\partial v}{\partial U} + BVY \frac{\partial v}{\partial V} \right\} A +$$

$$+ u_{fe} b_e \cos \alpha_e + v_{fn} b_n \cos \alpha_n -$$

$$- u_{fw} b_w \cos \alpha_w - v_{fs} b_s \cos \alpha_s = 0, \tag{6}$$

$$u_f = -\Delta t_f \left(EUU \frac{\partial P}{\partial U} + \beta g_U \Delta T \right) , \tag{7}$$

and:

$$v_f = -\Delta t_f \left(FVV \frac{\partial P}{\partial V} + \beta g_V \Delta T \right) , \tag{8}$$

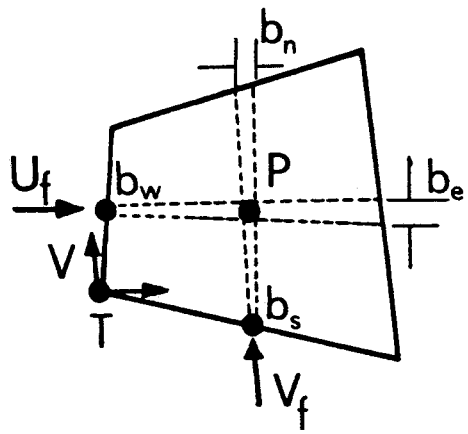

Fig. 1. A computational cell illustrating the centering of variables

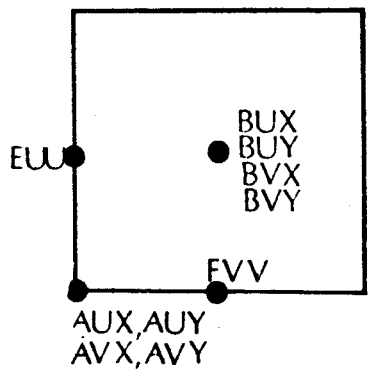

Fig. 2. A cell in the reference plane showing the centering for the coefficients of the transformation.

where: $\Delta t_r = K_r / v_f$

and: $\Delta t_f = K_f / v_f$ (9)

In equations 9, K_r or K_f must be a function of the location, if the permeability of the rock is variable or if the aperture of the fracture varies, respectively.

The various coefficients in equations 4-8 measure the influence of the coordinate transformation upon the original equations. They are numerically calculated as follows:

$AUX = U_x$ $\qquad$ $AUU = U_x^2 + U_y^2$

$AVX = V_x$ $\qquad$ $AVV = V_x^2 + V_y^2$

$AUY = U_y$ $\qquad$ $EUU = (AUX.AVY-AVX.AUY)/\sqrt{AVV}$

$AVY = V_y$ $\qquad$ $FVV = (AUX.AVY-AVX.AUY)/\sqrt{AUU}$

$gU = -g.AVX/\sqrt{AVV}$

$gV = g.AUX/\sqrt{AUU}$

and, finally:

$$\cos\alpha = (AUX.AVY-AVX.AUY)/(\sqrt{AUU}\ .\sqrt{AVV})$$

In equation 6, A is the area of the porous block. For the actual calculations, A was considered as the area of the cell without discounting the area of the fracture. This fact introduced an error of the orden of b.u, where u is the value of a typical velocity in the bulk of the rock. Taking into account the small aperture of the fracture with respect to a typical cell face dimension and that $u \ll u_f$, this error was negligible.

Figure 2 shows the location of the points for the calculation of the different coefficients of the coordinate transformation. Coefficients BUX have the same meaning as AUX and they only differ in their location at the grid. It is important to point out that the calculation of these coefficients -of course -greatly influences the global accuracy of the computed solutions. It is recommended that the grid transformation should also include the mid-points. In "smooth" grid transformations, the criterion adopted was to employ suitable averages of a basic set of coefficients (the ones denoted as AUX). In cases of greatly distorted grids, this criterion should be exercised very carefully.

The discrete version of equations 4-8 is obtained by means of centered-difference expresions in the computational plane, implying an appropriate averaging of the variables in the cell. They are as follows:

$$u_{ij} = \Delta tr \{ AUX_{ij} (P_{i-1j-1} + P_{i-1j} - P_{ij-1} - P_{ij})/2 + + AVX_{ij} (P_{i-1j-1} + P_{ij-1} - P_{i-1j} - P_{ij})/2 \} ; \quad (10)$$

$$v_{ij} = +\Delta tr \{ AUY_{ij} (P_{i-1j-1} + P_{i-1j} - P_{ij-1} - P_{ij})/2 + + AVY_{ij} (P_{i-1j-1} + P_{ij-1} - P_{i-1j} - P_{ij})/2 - - \beta g \Delta T_{ij} \} ; \quad (11)$$

$$D_{ij} = \{ (u_{i+1\ j+1} + u_{i+1j} - u_{i\ j+1} - u_{ij}) . BUX_{ij}/2 + + (u_{i+1\ j+1} + u_{ij+1} - u_{i+1j} - u_{ij}) . BVX_{ij}/2 + + (v_{i+1\ j+1} + v_{i+1j} - v_{ij+1} - v_{ij}) . BUY/2 + + (v_{i+1\ j+1} + v_{ij+1} - v_{i+1j} - v_{ij}) . BVY/2 \} A_{ij} + + (b.v_f.\cos\alpha)_n - (b.v_f.\cos\beta)_s + + (b.u_f.\cos\alpha)_e - (b.u_f.\cos\alpha)_w = 0 \quad (12)$$

$$u_{fij} = \Delta t_f \{ EUU_{ij} (P_{i-1j} - P_{ij}) - \beta\, gU_{ij+1/2}\, \Delta T_{ij+1/2} \}, \quad (13)$$

and:

$$v_{fij} = \Delta t_f \{ FVV_{ij} (P_{ij-1} - P_{ij}) - \beta\, gV_{i+1/2j}\, \Delta T_{i+1/2j} \}. \quad (14)$$

In expressions 13 and 14, the centering of T at the points in the middle of the cell faces was denoted with half indexes.

From now on the differences between the algorithms must be pointed out. The iterative method will be considered first. The algorithm closely resembles the one sketched in /5/ but, for the sake of completeness, it will be specified here. The calculation starts with a guessed pressure field, giving approximate values for u_{ij}, v_{ij}, u_{fij} and v_{fij} (from equations 10, 11, 13 and 14). Then the following iterative procedure is applied:

i) The discrete analogue of the cell divergence, D_{ij}, is obtained from equation 12.

ii) The pressure in the cell is adjusted to bring cell divergence to zero, as follows:

$$P_{ij}^{(k+1)} = P_{ij}^{(k)} + \delta P_{ij}$$

where:

$$\delta P_{ij} = -\beta_{ij} D,$$

$$\beta_{ij} = \frac{\omega}{\Delta t_r A_{ij}}\{ BUU_{ij} + BVV_{ij} + CT \}^{-1} +$$

$$+ \frac{\omega}{\Delta t_f}\{ (b \cos\alpha\ FVV)_n + (b \cos\alpha\ FVV)_s + (b \cos\alpha\ EUU)_e +$$

$$+ (b \cos\alpha\ EUU)_w \}^{-1}$$

and CT means Cross Terms (i.e., higher order terms implying crossed derivatives of the transformation coefficients).

ω is an over relaxation parameter whose typical value is 1.7 for this type of problems.

iii) The velocity of components at the cell corners are accordingly modified as follows:

$$u^{(k+1)}_{i+1j+1} + u^{(k)}_{i+1j+1} + C\ (AUX_{i+1j+1} + AVX_{i+1j+1}),$$

$$u^{(k+1)}_{i+1j} + u^{(k)}_{i+1j} + C\ (AUX_{i+1j} - AVX_{i+1j}),$$

$$u^{(k+1)}_{ij+1} + u^{(k)}_{ij+1} + C\ (-AUX_{ij+1} + AVX_{ij+1}),$$

$$u^{(k+1)}_{ij} + u^{(k)}_{ij} + C\ (-AUX_{ij} - AVX_{ij}),$$

$$v^{(k+1)}_{i+1j+1} + v^{(k)}_{i+1j+1} + C\ (AUY_{i+1j+1} + AVY_{i+1j+1}),$$

$$v^{(k+1)}_{i+1j} + v^{(k)}_{i+1j} + C\ (AUY_{i+1j} - AVY_{i+1j}),$$

$$v^{(k+1)}_{ij+1} + v^{(k)}_{ij+1} + C\ (-AUY_{ij+1} + AVY_{ij+1}),$$

$$v^{(k+1)}_{ij} + v^{(k)}_{ij} + C\ (-AUY_{ij} - AVY_{ij})$$

$$u^{(k+1)}_{fi+1j} + u^{(k)}_{fi+1j} + C_f\ EUU_{i+1j},$$

$$u^{(k+1)}_{fij} + u^{(k)}_{fij} - C_f\ EUU_{ij},$$

$$v^{(k+1)}_{fij+1} + v^{(k)}_{fij+1} + C_f\ FVV_{ij+1},$$

and

$$v^{(k+1)}_{fij} + v^{(k)}_{fij} - C_f\ FVV_{ij},$$

In the previous expressions C and C_f are defined as:

$$C = \Delta t_r \quad \frac{\delta P_{ij}}{2} \quad ; \quad C_f = \Delta t_f \; \delta P_{ij}$$

Boundary conditions were appropriately applied after each iteration sweep and convergence was usually reached. This convergence and subsequent pressure smoothing defined the complete solution (see /6/).

As presented, the iterative algorithm solves Poisson's pressure equation through the use of an intermediate variable (D) and provides a simultaneous adjustment of velocities. An alternative approach consists in solving this Poisson's equation from the very beginning; this procedure was also implemented and it is, sometimes, less sensitive to steep variations in pressure, allowing, in turn, for a simpler treatment of "singularities" in the sense defined in /8 , 14/.

The other algorithm implied the use of a direct solver in order to obtain a fully coupled solution for equations 4-8. The equations were ordered by blocks, each one representing a computational cell. In each block, the equations were ordered as follows.

i) u momentum (from which u was obtained)

ii) continuity (from which v was obtained)

iii) v momentum (from which P was obtained)

iv) u_f momentum (from which u_f was obtained)

v) v_f momentum (from which v_f was obtained)

Boundary conditions were incorporated explicitly in the system of equations.

The definition of the coefficients in these equations is given in the Appendix for the general blocks.

The algebraic system of linear equations was solved with a library sparse matrix solver named MA28AD from the Harwell package /9/, which proved to be efficient in most cases. However, attention must be paid to the relative weight of the coefficients if meaningful results are desired. The resulting size of the systems of algebraic equations was a function of the nodes employed and considerable fill-in arose.

The treatment of boundary condition was, of course, determinant for the obtention of results. Several types of them may be prescribed, namely:

i) inflow-outflow boundaries, with both rock and fracture fluid velocities specified;

ii) free-slip boundaries, where the normal component of the velocity vector is null. In this case the velocity at the boundary is obtained from the velocity in an inner point as in /6/, i.e.:

$$\bar{u}_B = (\bar{u}_I \cdot \bar{n})\,\bar{n} - \bar{u}_i \ ,$$

where I and B represent the Boundary and the Inner points respectively and $\bar{n}$ is a unit vector normal to the boundary;

iii) boundaries with imposed pressure, where the velocities are imposed as continuative and the pressures are a function of the space coordinates;

iv) boundaries with continuative velocity, where the velocity is specified as a function of the interior one;

v) periodic boundaries, where the inflow and outflow are linked by periodicity.

When dealing with the iterative method, a greater flexibility is allowed for the imposition of boundary conditions. It is known (see /10/ for example) that iterative methods give a solution even if the pressure equation is over-specified with respect to boundary conditions. In the case of the direct method this flexibility is, of course, not allowed.

The inherent checkerboarding pressure field is a consequence of the type of variable centering adopted and this fact is also related with the specification of boundary conditions. The presence of the fractures seemed to change the patterns of the checkerboard modes and, therefore, the simple smoothing technique of /6/ must be employed carefully. A smoothing technique of general validity is still under research. Actually, the pressure field is not of substantial importance in the case of steady flows when dealing with problems of transport of solutes. However, in the unsteady-flow case, the smoothing of the cell's pressure at the end of a time step affects the estimation of the velocity for the next step.

In the direct method, the imposition of boundary conditions also implied the reordering of equations in the cells adjascent to the physical boundaries. This was not the case with the iterative procedure.

RESULTS AND DISCUSSION

In this section the results of the simulation of three typical cases are shown and the relative merits of both algorithms are discussed.

The first case was taken from reference /11/ and consists in predicting the flow in a rectangular network of intersecting fractures in a porous media. The problem is linear and this fact allowed the comparison of results from the data in /11/.

The geometry of the network is shown in figure 3. When the permeability of the porous media is some orders of magnitude lower than the corresponding permeability of the fractures, this network resembles a net of conduits with linear flow resistance (as pointed out in /11/). The conduits were uniform in length (200 m) and in aperture (10^{-3} m).

The flow was established by imposing a constant flow boundary condition to the fractures on the left face of the block and by extracting fluid at a constant flow rate from one fracture. Since the model considers a porous fractured rock, this situation can be modelled when the corresponding permeability in the porous matrix is far lower than that of the fractures. $K_r = 10^{-30}$ m^2 (almost zero) was adopted. The results obtained were compared with the ones in reference /11/ and both are shown in figure 3 in a "non-dimensional" form. This presentation was adopted because an exact comparison was not feasible with the data reported in reference /11/.

The results compare fairly well; differences are below 4%, except in the first column of nodes, where they reached 8%. This column is an additional one, and was only necessary in order to impose input flow conditions. The results were checked for sensitivity to the values of K_r. No differences were found in the range $10^{-3} \leq |b| \leq 10^{-5}$, with b measured in m.

The second case consists in the modelling of the flow around a circular cilinder and is representative of the situation found in a plane normal to the axis of a nuclear waste container placed in a fully saturated rock. In this case the flow resembles the potential flow around a circular cylinder. The grid adopted is the one shown in figure 4a.

The boundary conditions were as follows:

- Prescribed flow on the external boundary, as given by the theoretical solution
- Free slip rigid wall on the cylinder boundary.
- Periodic flow conditions along segment A-B.

The resulting flowfield is represented in the vector plot shown in figure 4b. A close up for a finer grid calculation is shown in figure 4c. The maximum error in the case of figure 4b was 5% with respect to the analytical solution. The results are very good in spite of these coarse grids. This is coherent with the results of reference /12/, where accurate results were obtained with 4/1 elements in a Navier-Stokes FEM code. Quite surprisingly, elements of a higher order produced spurious velocity oscillations all over the flow-field.

The third example consists in predicting the flow towards an inclined fracture. Figure 5 illustrates the grid employed and the fracture. The domain of integration is now a square block of homogeneous rock, 1000 m high and 1000 m wide, the permeability being fixed at 1.7×10^{-15} m^2 (a typical value for a cristalline rock). The inclined fracture ran all across the

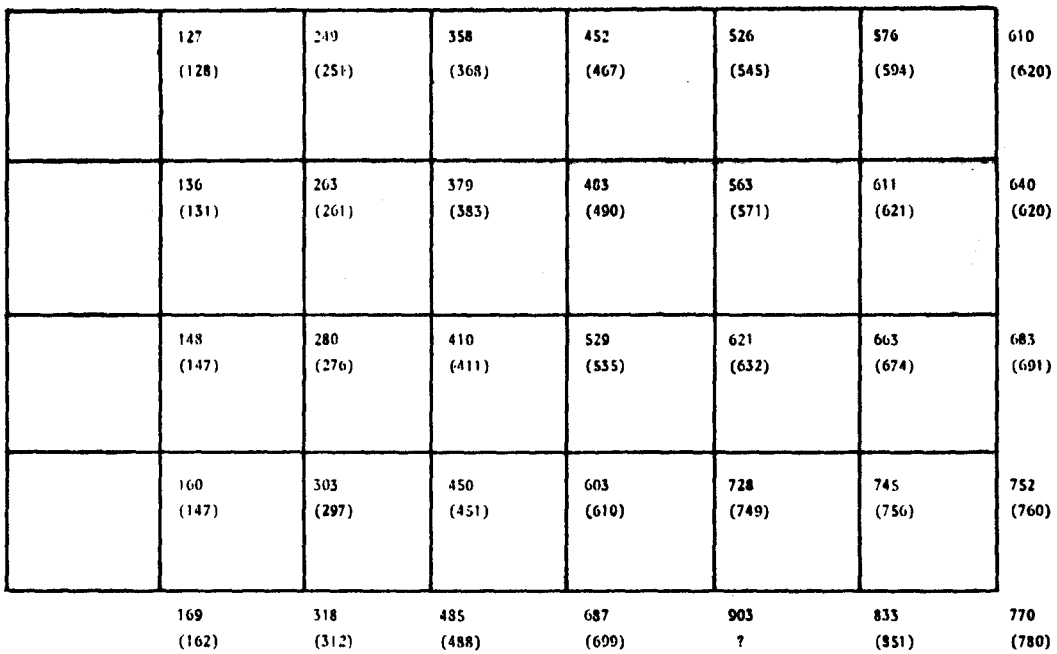

Fig. 3. A network of fractures of equal size /11/, $b=10^{-3}$ m.
Comparison of results for $(P_{ij} - P_o)/(P_I - P_o)$.
Values between parenthesis taken from /11/

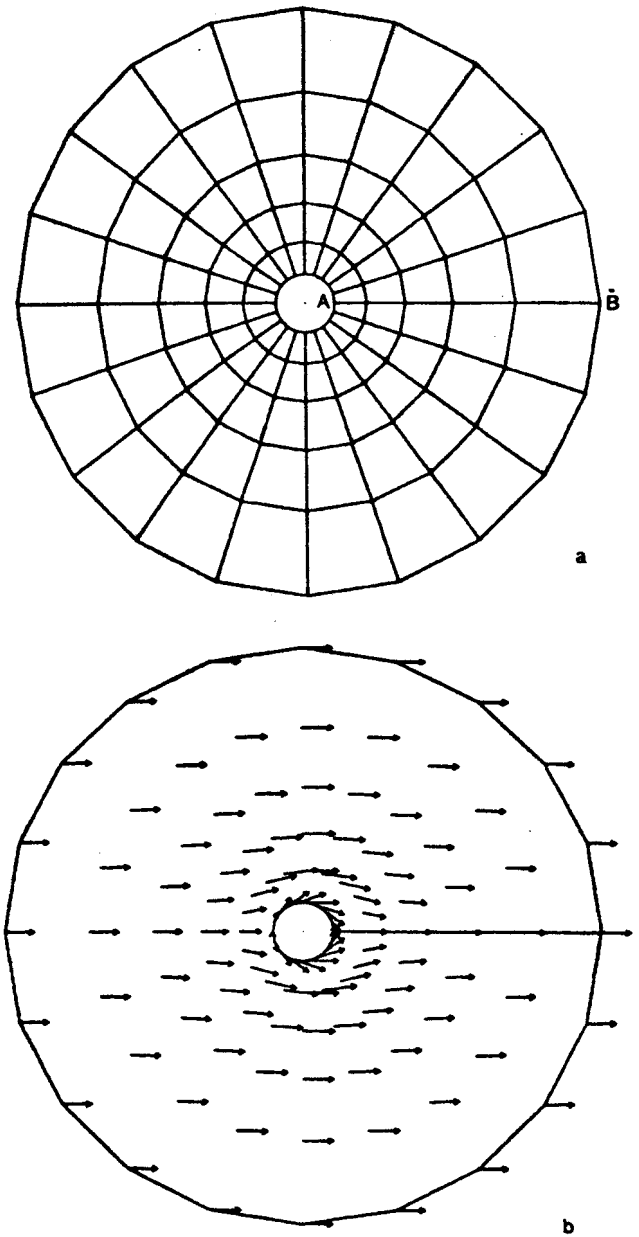

Fig. 4. Flow around a circular cylinder. (a) grid, (b) vector plot.

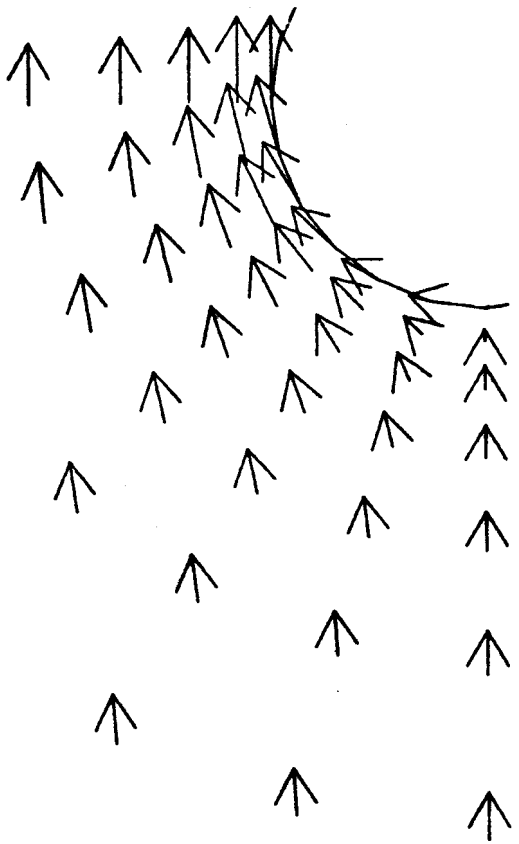

Fig. 4. Flow around a circular cylinder. (c) Partial vector plot for a finer grid.

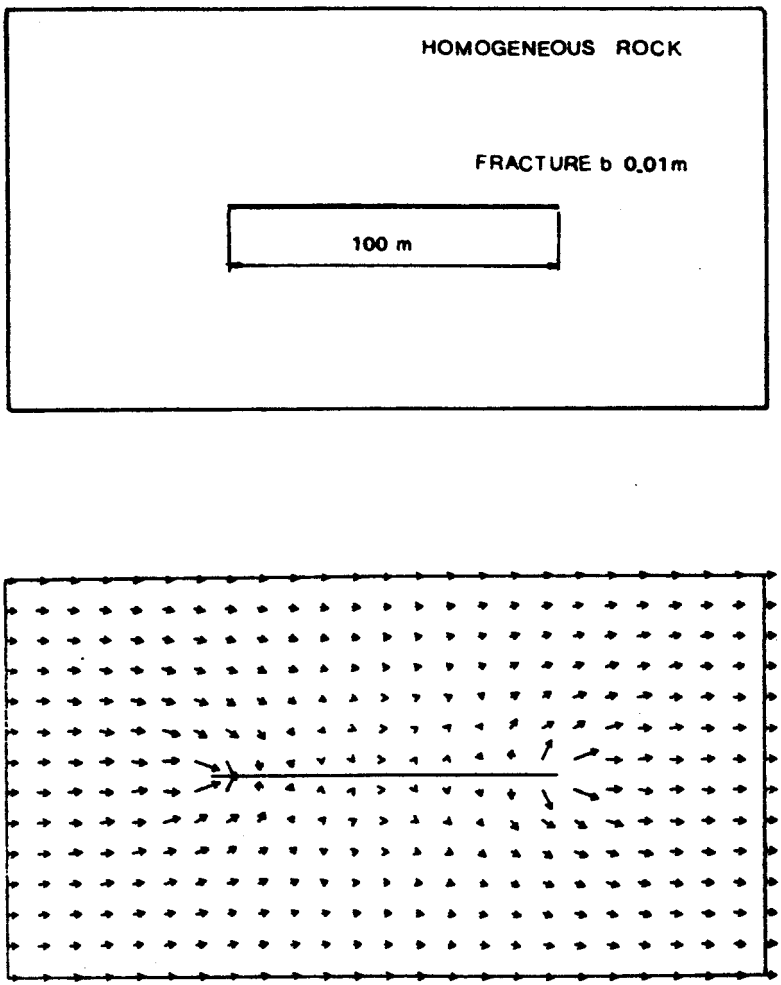

Fig. 6. Flow past an isolated fracture in a homogeneous rock. (a) Domain of integration, (b) vector plot.

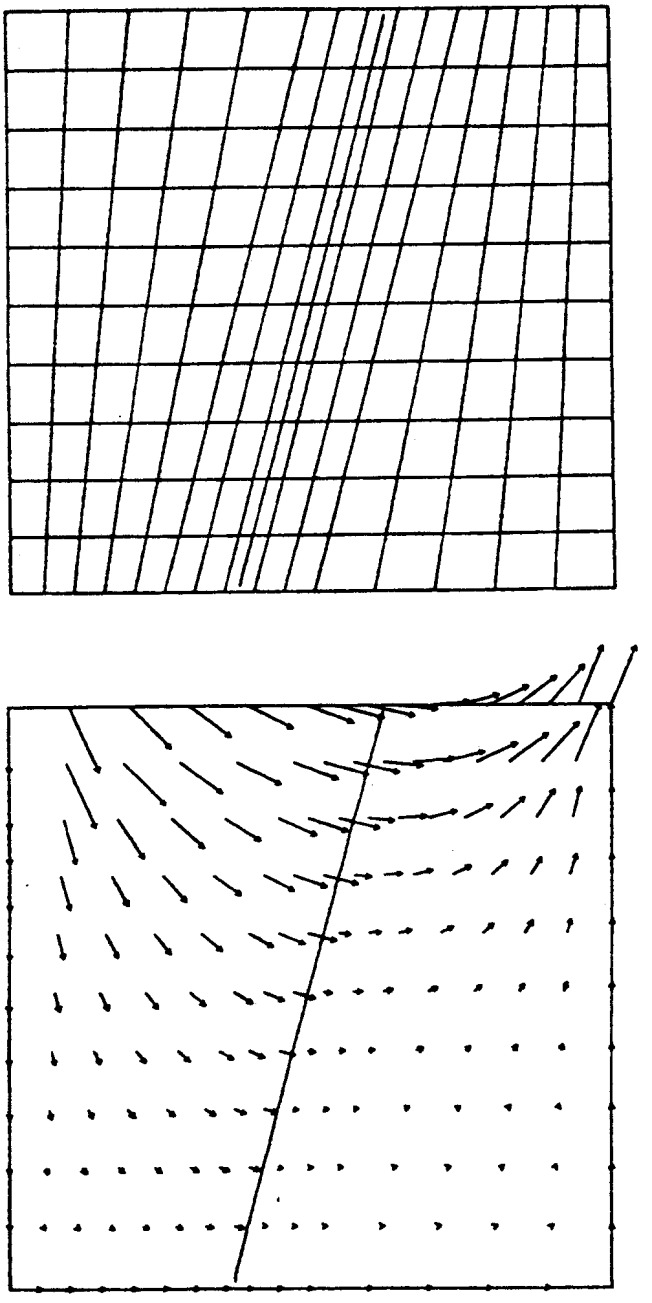

Fig. 5. Flows across an inclined fracture. (a) grid, (b) vector plot. $b = 10^{-2}$ m.

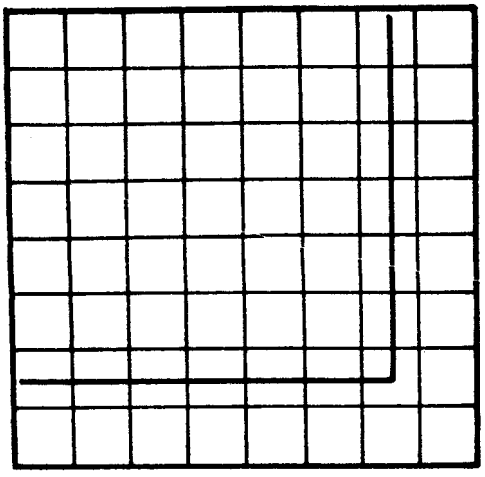

Fig. 7. A grid for the comparison of results for a single fracture. /15/

No. cells	DIRECT METHOD		ITERATIVE METHOD	
	memory (kB)	TCPU (sec)	memory (kB)	TCPU (sec)
7	21	0.75	8	0.52
11	43	1.52	15	3.40
15	73	3.05	24	13.3
21	135	7.35	42	50.1
31	296		84	
41	>1024		139	490

Table 1 - Comparison of the resources involved for a given problem.

DIRECT METHOD		ITERATIVE METHOD	
memory (kB)	TCPU (sec)	memory (kB)	TCPU (sec)
172	18.3	48	129

Table 2 - Comparison of the resources involved for the problem in figure 5. /11/

		E %
Primitive variables	direct solver	2.0
	iterative solver	2.0
Pressure	direct solver	-1.1
	iterative solver	-1.0

Table 3 - Comparison of the maximum error in the fracture velocity for the case in figure 5.

rock and had a permeability of 1.7×10^{-9} m^2 with b = 0.01 m (linear law).

The boundary conditions were as follows:

- Impermeable walls ($u_n = u_f = 0$) at the right, left and bottom boundaries.
- Imposed water table (given pressure) varying linearly from left to right with a gradient of 3.87 m/s^2.
- Continuative (open) fracture at the top boundary.

The resulting flowfield is shown in figure 5b as a vector plot. As may be seen, the flow is almost interrupted when the fluid reaches the fracture which, in turn, takes up most of the incoming fluid; the resulting velocity in the fracture was 2.44 10^{-3} m/s.

The last example was also extracted from reference /11/ and consists in the prediction of the flow in a homogeneous porous media with an isolated fracture parallel to the velocity field at infinite distance. Figures 6a and 6b show the computational domain and a vector plot of the resulting flow field. The results were obtained without imposing simmetry conditions with a grid of 25 x 14 nodes. Boundary conditions were imposed at "infinity" as given by the analytical solution in reference /13/. It is interesting to point out that the grid was coarse and that the solution was only obtained after applying a technique similar to that in reference /14/. Some details of this technique will be given elsewhere. Permeabilities were $K_r = 10^{-15}$ m^2 and $K_f = 10^{-7}$ m^2 for the homogeneous rock and the fracture, respectively. The maximum apparture of the fracture was 10^{-2} m (linear law). The results obtained compared fairly well with the analytical ones. Errors were in the order of 2% in the immediate vicinity of the fracture when compared with the analytical solution and tended to zero towards the limits of the domain.

The above cases served to exemplify the capabilities of the codes implemented. Thus, this is the right time to perform some comparisons among the relative merits of both algorithms. Computational efficiency is a measure of a code's capability to perform a given task and usually involves the so-called "grind time", i.e. the cost in seconds of CPU divided by the number of cells multiplied by the number of iteration sweeps through the mesh required to reach a given error. This is a simple measure but does not make any reference to the resources needed for the actual calculations. A better (more realistic) cost would be that obtained by considering all the resources involved. However, the first definition is usually found in literature and is the one applied herewith.

Some tests were performed against a one-dimensional problem as given in reference /15/. The codes were considered as fully two-dimensional and the physical situation is the one schematically shown in figure 7. Table 1 shows the different costs for a BASF 68 computer, as a function of the number of

cells for the one-dimensional path. As may be seen in this table, the direct method is more economical in CPU time but far more expensive in memory. The economy in CPU time is sometimes problem dependent because, if additional time steps are necessary, the iterative code is cheaper because the initial values are closer to the final ones after the first time step (actually it was the case in the calculations of reference /5/).

For the problem in figure 3, the results are shown in table 2, confirming the trends in table 1. It is obvious from the preceeding tables that the grind time is much shorter for the direct method.

In spite of the previous conclusion, the iterative method was, as stated in /5/, partially abandoned because the authors felt that the direct method was more reliable. However, in order to test new improvements to the codes, the iterative method was later (once again!) preferred, because changing the programming of the direct code was not an easy task. The direct solver implies another shortcoming in the sense that a parameter must be selected to govern the pivoting. Improper setting implied a greater growth of the matrix and sometimes the doubt remained regarding the validity of the solution so obtained. The iterative solver is, as discussed previously, less sensitive to an over-specification of the problem, this characteristic being (in the authors' opinion), desirable.

As a last example, the performance of the codes in terms of pressure were compared with the primitive variable versions, for the case of figure 6 /11/. The comparison was performed in terms of the maximum error in velocity at the fracture.

In this case a very coarse grid was employed (12x9 cells), imposing analytical boundary conditions at a distance of the order of 70 m from the fracture. The errors in velocity at the fracture are shown in table 3 for the various versions and are, of course, similar.

The extension of the present method to three space dimensions and unsteady flow (now under research) imposes limitations due to the limited computer resources available and, as a (definitive?) rule, the iterative method will be preferred.

CONCLUSIONS

Two different algorithms for the prediction of hydrodynamics in fractured-porous media have been presented. Their relative merits have been discussed in terms of the results obtained from their associated codes for a set of verification problems. It may be concluded that all versions are of similar accuracy. However, from the point of view of their exploitation, the version considering the direct solver is the most adequate for steady-state problems. In new situations and, particularly, in three-dimensional flow, the iterative version is simpler to implement and far more economical in computer memory. The same conclusion applies to unsteady flow situations.

The codes allowed for the inclusion of discrete fractures in a domain subdivided in cells. This approach is rare in computational methods and is particularly suitable for the inclusion of semi-analytical solutions toward improving the global accuracy of the results.

REFERENCES

/1/ Ene, H.I. and Sánchez-Palencia, E., "On thermal equation for flow in porous media", Int. J. Eng. Sci., 20, No.5, pp. 623-630, 1982.

/2/ Anonimous, Proceedings of "Workshop of Thermohydrological Flows in Fracture Rock Masses, LBL-11566 (ONWI 240), February 19-20, 1980, Berkeley, California, USA.

/3/ Narasimhan, T.N., "Multidimensional Numerical Simulation of Fluid Flow in Fractured Porous Media", Water Resources Res., 18, No.4, pp. 1235-1247, 1982.

/4/ Pruess, K and Narasimhan, T.N., "A Practical Method for Modelling Fluid and Heat Flow in Fractured-Porous Media, J. of Soc. Pet. Eng., pp. 14-26, February, 1985.

/5/ Ferreri, J.C. and Grandi, G.M., "Models for the Study of Local Effects Produced by a High-Level Radioactive Waste Repository", in Numerical Methods in Laminar and Turbulent Flows, C. Taylor, M.D. Olson, P.M. Gresho and W.G. Habashi (Editors), Pineridge Press, pp. 1257-1267, 1985

/6/ Martínez, B. and Ferreri, J.C., "SOLA-BFC A computer code for unsteady fluid flow calculations with boundary-fitted coordinates", Annals of the III Congreso Latinoamericano sobre Métodos Computacionales en Ingeniería A. Ferrante (Editor), Buenos Aires, pp. 614-630, 1982.

/7/ Thompson, A.F., "Numerical solution of flow problems using boundary-fitted coordinate systems," in Computational Fluid Dynamics, W. Hollman (Editor), Hemisphere, pp. 1-98, 1980.

/8/ Ferreri, J.C. and Ventura, M.A., "Numerical Aspects of the Study of the Thermal Regional Impact of a Radioactive Waste Repository", to appear in Nuclear Engineering and Design, 1985.

/9/ AERE, "Harwell subroutine library - A catalogue of subroutines", compiled by M.J. Hoofer, AERE-R9185, 1980.

/10/ Patankar, S.V., "Numerical Heat Transfer and Fluid Flow", Hemisphere, 1980.

/11/ Baca, R.G., Arnett, R.C. and Langford, D.W., "Modelling fluid flow in fractured-porous rock masses by finite-element techniques", Int. J. Num. Methods in Fluids, 4, pp. 337-348, 1984.

/12/ Lee, R.L., Gresho, P.M. and Sani, R.L., "An exploratory study on the application of an existing finite element Navier-Stokes code to compute potential flows", Int. Conf. on Finite Element Methods, Shangai, China, August 2-6, 1982.

/13/ Strack, O.D.L., "Analytic modelling of flow in a permeable fissured medium", PNL 4005, Pacific Northwest Laboratory, Richland, Washington, 1981.

/14/ Ferreri, J.C., "A note on the steady-state advection-difussion equation", Int. J. Numerical Methods in Fluids, to appear, 1985.

/15/ Wang, J.S.Y., Tsang, C.F., Cook, N.G.W. and Witherspoon, P.A., "A study of regional temperature and thermohydrologic effects of an underground repository for nuclear wastes in hard rock", J. of Geophysical Res., 86, pp. 3759-3770, 1981.

A P P E N D I X

EQUATION	DIAG. TERM.	D E F I N I T I O N S
11	*	$A = \Delta t_r\ (AUX_{ij} + AVX_{ij})/2$ $A_2 = \Delta t_r\ (-AUX_{ij} + AVX_{ij})/2$ $A_3 = \Delta t_r\ (AUX_{ij} - AVX_{ij})/2$ $A_4 = 1.0$ $A_5 = \Delta t_r\ (-AUX_{ij} - AVX_{ij})/2$
12	*	$B_1 = A_{ij}\ (BUX_{ij} + BVX_{ij})$ $B_2 = A_{ij}\ (BUY_{ij} + BVY_{ij})$ $B_3 = 2\ b_w \cos \alpha_w$ $B_4 = 2\ b_s \cos \alpha_s$ $B_5 = A_{ij}\ (BUX_{ij} - BVX_{ij})$ $B_6 = A_{ij}\ (BUY_{ij} - BVY_{ij})$ $B_7 = -\ 2\ b_e \cos \alpha_e$ $B_8 = A_{ij}\ (-BUX_{ij} + BVX_{ij})$ $B_9 = A_{ij}\ (-BUY_{ij} + BVY_{ij})$ $B_{10} = -\ 2\ b_n \cos \alpha_n$ $B_{11} = A_{ij}\ (-BUX_{ij} - BVX_{ij})$ $B_{12} = A_{ij}\ (-BUY_{ij} - BVY_{ij})$
13	*	$C_1 = \Delta t_r\ (AUY_{ij} + AVY_{ij})/2$ $C_2 = \Delta t_r\ (-AUY_{ij} + AVY_{ij})/2$ $C_3 = \Delta t_r\ (AUY_{ij} - AVY_{ij})/2$ $C_4 = 1.0$ $C_5 = \Delta t_r\ (-AUY_{ij} - AVY_{ij})/2$
14	*	$D_1 = -\ \Delta t_f\ (EUU_{ij})$ $D_2 = -\ D_1$ $D_3 = 1.0$
15	*	$E_1 = -\ \Delta t_f\ (EVV_{ij})$ $E_2 = -\ E_1$ $E_3 = 1.0$