Title

CHANGE: A Numerical Model for Three-Dimensional Modelling of Channelized Flow in Rock. Theory and Design

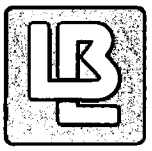
Authors

Billaux, D.
Long, J.C.S.
Peterson, J.E.

Publication Date

1990-03-01

LBL-24910
UC-403



Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

EARTH SCIENCES DIVISION

CHANGE: A Numerical Model for Three-Dimensional Modelling of Channelized Flow in Rock. Theory and Design

D. Billaux, J.C.S. Long, and J.E. Peterson, Jr.

March 1990

Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

DISCLAIMER

LBL-24910

CHANGE: A Numerical Model for Three-Dimensional Modelling of Channelized Flow in Rock. Theory and Design

Daniel Billaux, Jane C. S. Long, and John E. Peterson Jr.

Earth Sciences Division
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

March 1990

This work was supported by the Manager, Chicago Operations, Repository Technology Program, Repository Technology and Transportation Division, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

ABSTRACT

A model for channelized flow in three-dimensional, random networks of fractures has been developed. In this model, the fractures are disc-shaped discontinuities in an impermeable matrix. Within each fracture, flow occurs only in a network of random channels. The channels in each fracture can be generated independently with random distributions of length, conductivity, and orientation in the fracture plane. Boundary conditions are specified on the sides of a "flow region", and at the intersections of the channels with interior "holes" specified by the user to simulate boreholes or drifts. This code is part of a set of programs used to generate two-dimensional or three-dimensional random fracture networks, plot them, compute flow through them and analyze the results.

Table of Contents

List of Figures

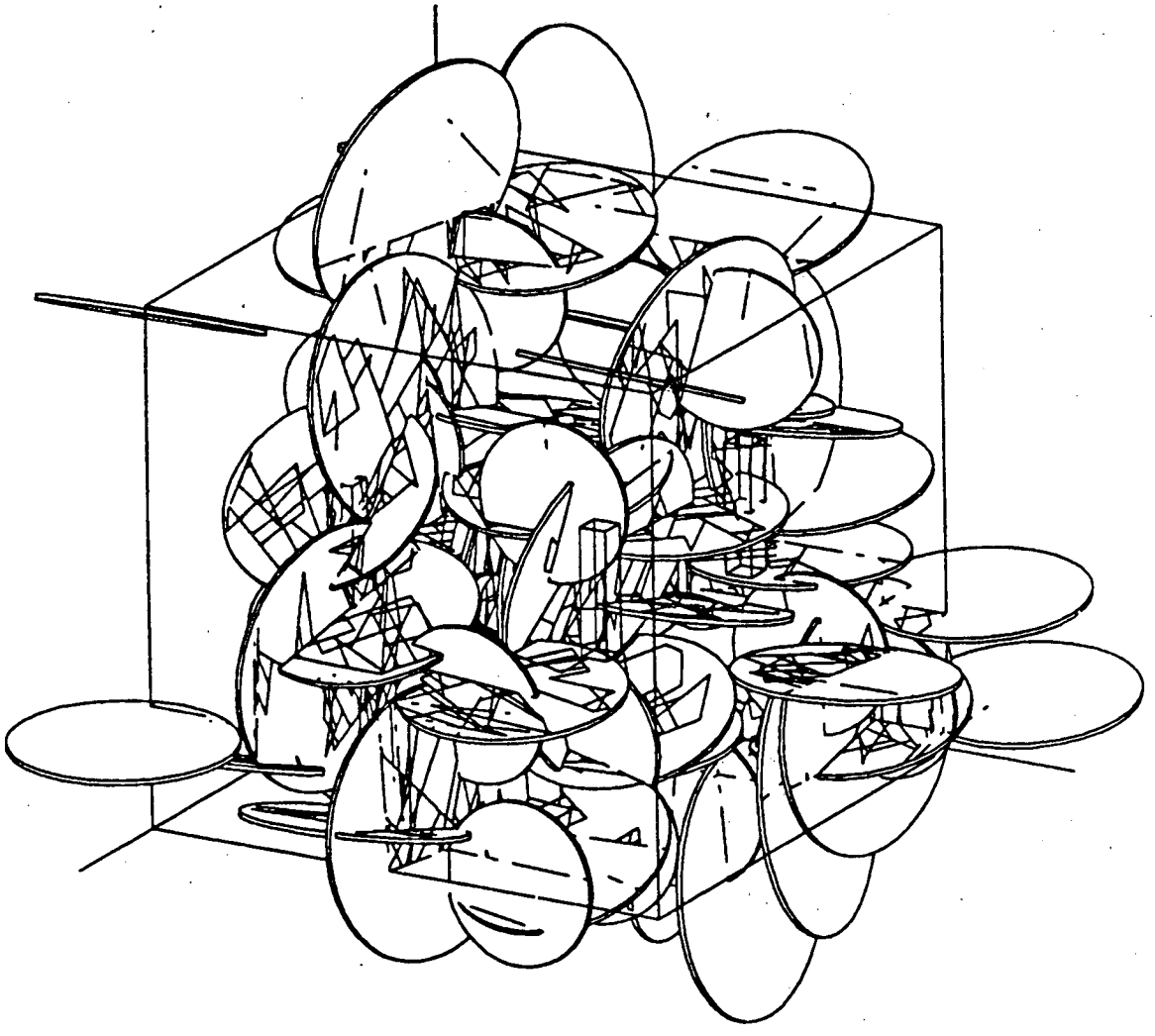
1.0 INTRODUCTION

Two-dimensional and three-dimensional stochastic models of fracture flow have been built at LBL during the last five years. These models rested on the assumption that the flow in a given fracture is uniform and can be approximated as flow through a parallel-plate or a slab of porous media (Long, 1983; Gilmour et al., 1986). However, there is now evidence that the voids in fractures often form channels (Gentier, 1986; Neretnieks et al., 1987). These channels can be considered as a mesh of quasi-one dimensional channels. Building upon our previous experience with two and three-dimensional fracture networks, we have designed and coded a new program, CHANGE (CHANnel GEnerator), which generates random channels on a given network of discs and outputs a three-dimensional finite-element grid of line elements.

1.1 Channel Generation

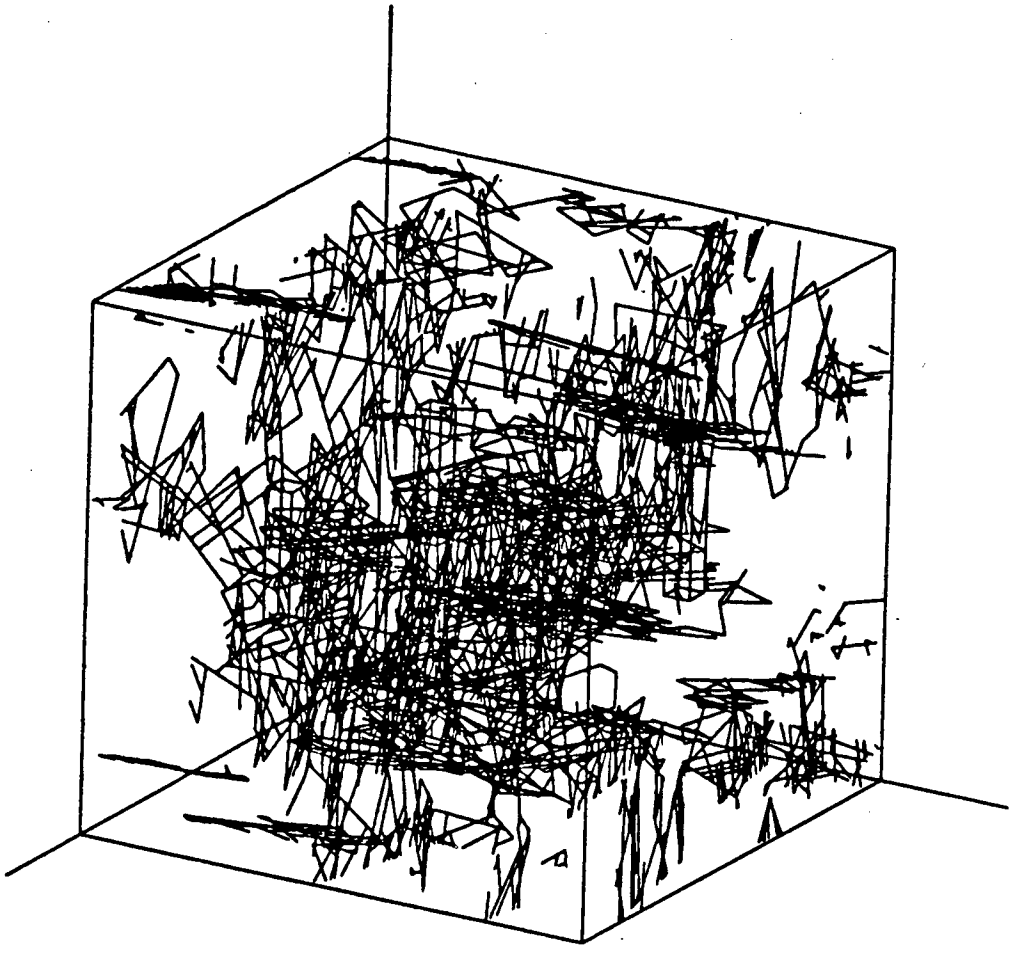
Input to the program are the complete specification of a three-dimensional network of discs as produced by program FMG3D (Gilmour et al., 1986a and b), the statistical properties needed for the channel network on the discs, and boundary conditions. The channels in each fracture can be generated independently with random distributions of orientation in the fracture plane. Complex distributions of channel characteristics can be obtained by superposition of several channel sets on a same fracture disc. If the fracture discs have been generated in several sets, then the characteristics of the channels can be controlled in each fracture set independently. In this way, different fracture morphologies can be reproduced in the same rock mass. Figure 1-1 shows an example of a three-dimensional network of disc-shaped fractures with sub-networks of channels in the fractures. In Figure 1-2, only the channels network is plotted.

We currently generate channels over the whole area of each fracture disc. However, the generation of channels within a fracture disc could be confined to any sub-area of the disc. This



XBL 881-324

Figure 1-1. Part of a mesh of channels on discs. The broken lines are intersections with discs not shown.



XBL 881-325

Figure 1-2. Channels only from mesh in Figure 1-1.

would effectively eliminate the constraint of having disc shaped fractures. In fact, the disc shape is necessary only to find the intersections between fractures.

1.2 Pipe Network in Flow Region

The region in which flow can be studied must be a rectangle parallelepiped with any orientation in space. In addition, any number of rectangular-shaped "holes" can be specified within the region in order to simulate drifts, boreholes (Figure 1-3), or to get more complex outer boundary conditions. Before the mesh can be used for computing flow, the program must determine which channels are in the flow region, and which ones are truncated at the boundaries of the flow region. The boundary conditions will be applied to the endpoints of the truncated channels. The intersections between fracture discs are treated as a separate class of conductors in this model. The union of the generated channels and the fracture disc intersections makes up the flow mesh and we use the generic term of "pipe" for both kinds of conductors. Currently, the disc intersections are treated in the model as high conductivity channels. This seems to be geologically most reasonable. However, these intersections can be treated otherwise if it is indicated.

Once the mesh has been limited to the flow region, intersections between pipes must be determined. This is done from the boundaries inward, determining all the intersections between boundary pipes and other pipes, then intersections between these pipes and other ones, and so on. In this way, pipes not connected to a boundary are automatically discarded. An option can disable this feature and effectively produce a mesh containing all the pipes in the flow region, for pipe-matrix flow studies for example. If a pipe is a channel, then only other pipes in the same fracture disc can intersect it. If a pipe is a fracture disc intersection, it lays on two different discs, so the pipes on both discs must be searched for intersections. All the computations for truncating channels and finding pipe intersections are performed in the plane of the relevant fracture. This saves both computer memory and processing time. Once all intersections have been found, the program can discard simple dead-ends, i.e. pipe endpoints or pipes showing one intersection with the rest of the mesh. This option can be overridden for transient flow computations. The program outputs a mesh with pipe segments as line elements and pipe intersections as nodes, with

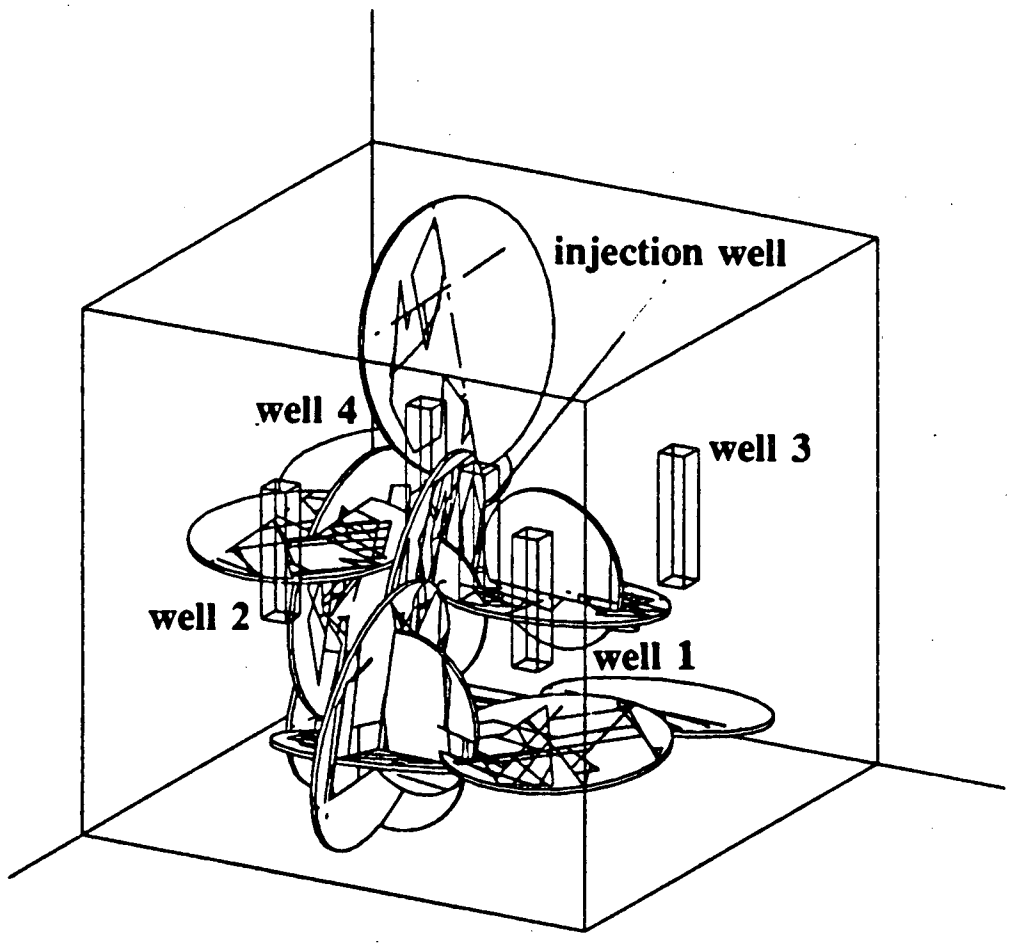


Figure 1-3. Modelling boreholes.

specifications of head or flux imposed at the boundary nodes.

1.3 Computing Flow

This mesh can then be processed by program RENUM (Billaux et al., 1988a and b) to remove dead-ends, shrink any given imposed flux boundary to a point, and renumber the nodes to minimize the bandwidth of the corresponding linear system of equations. Program LINEL (Billaux et al., 1988a and b) then computes steady-state flow in the mesh, and program TRINET (Karasaki, 1987) can compute transient flow and solute transport in the mesh. The newly completed chain of programs FMG3D, CHANGE, RENUM, LINEL and TRINET, is a unique tool for modelling flow and transport in very complex channelized fractured rock geometries (Figure 1-4).

A paper submitted to the international symposium on Hydrogeology and Safety of Radioactive and Industrial Hazardous Waste Disposal (Orléans, France, June 1988) is provided in an Appendix as an example use of the program.

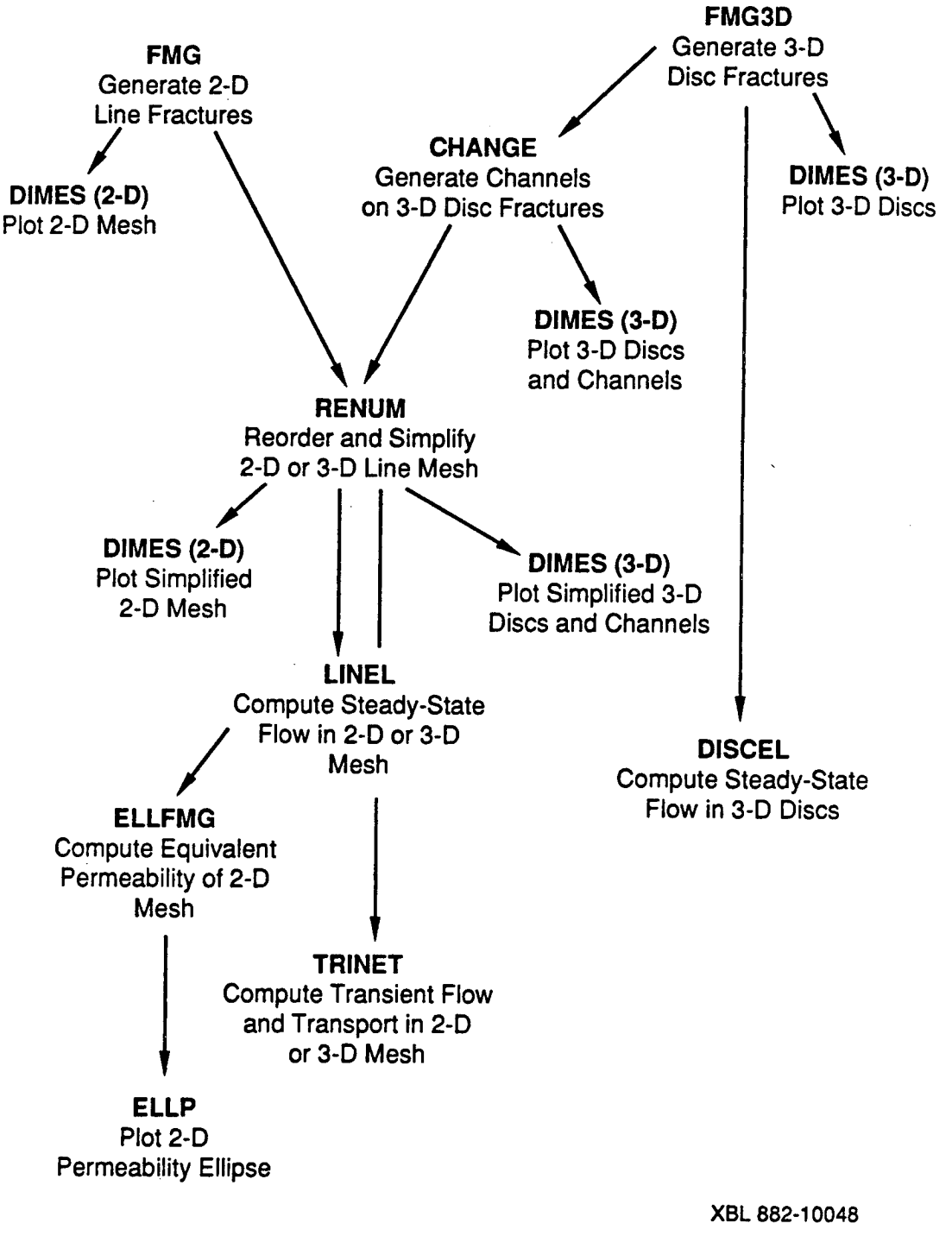


Figure 1-4. Organization of the set of programs.

2.0 GENERATION OF CHANNELS

On each circular disc generated by FMG3D, a channel system is created consisting of one or more sets of linear channels either randomly distributed or forming a grid within the disc.

2.1 Specification of the Channel System Characteristics

Two types of input are needed by the program in order to produce a system of channels. First a definition of the complete disc system generated by FMG3D (Gilmour et al., 1986) is read by subroutine RFRAC. Second the statistical distributions of the channel properties are read by subroutine RCHAN.

2.1.1 Definition of the Complete Disc System.

Subroutine RFRAC reads the description of individual fracture discs into array [frac(12,maxfrac)]. For each fracture, the program reads:

- the orientation angles phi and theta (Figure 2-1),
- the radius,
- the aperture,
- the coordinates of the center,
- the four coefficients of the equation of its plane

The set from which a disc was generated is also read into array [isetfr(maxfrac)].

2.1.2 Definition of the Statistical Distributions of the Channel Properties.

The channels can be divided into one or more sets. For each disc, one or more channel sets can be used to generate channels, depending on the fracture set from which the disc has been generated. The channel set(s) to be used for each fracture set are read into array [isetch(maxset,10)]. Then the characteristics for each channel set are read into arrays [ichsi(maxset,8)] and [rchsi(maxset,8)]. These characteristics include the areal density of channels on the discs, the type of distribution and distribution parameters for orientation, length, and transmissivity.

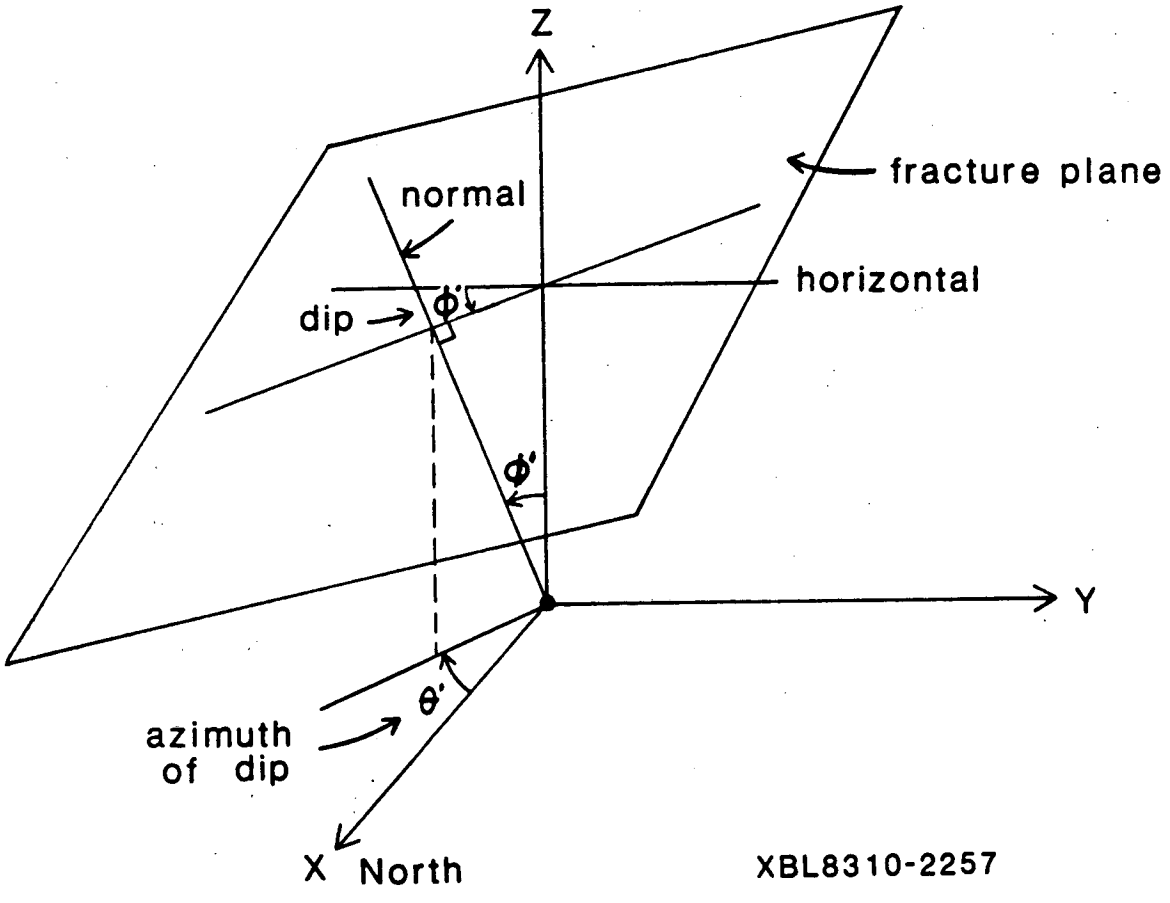


Figure 2-1. Orientation angles for fractures.

2.2 Generation of the Channels

Subroutine CHAGEN generates the channel characteristics using the specifications read by RCHAN. All the characteristics of the channels are generated for each fracture in turn. To facilitate this, the coordinates are rotated such that the new X-Y plane is parallel to the plane of the fracture, thus reducing the problem to two dimensions. This procedure is the same as the one used in FMG3D. Its description is repeated here for completeness.

2.2.1 Rotation of Axes

Subroutine ROFRAC computes a rotation matrix for a given fracture disc, with center at (x_c, y_c, z_c) lying in the plane $ax + by + cz + d = 0$, that will rotate the Z-axis such that it is normal to the plane. The new Z-axis, Z', has direction cosines a, b, and c relative to the XYZ system, intersects the plane at the point (-ad, -bd, -cd), and is at a distance -d from the origin. The X-axis is rotated such that it is parallel to a line through points (-ad,-bd,-cd) and (x_c, y_c, z_c) as shown in Figure 2-2a. The direction of this line is $(x_c + ad, y_c + bd, z_c + cd)$ and dividing by the distance, k, between the two points yields direction cosines l_1, l_2, and l_3 where

$$k = [(x_c + ad)^2 + (y_c + bd)^2 + (z_c + cd)^2]^{1/2},$$
$$l_1 = \frac{x_c + ad}{k}, \quad l_2 = \frac{y_c + bd}{k}, \quad l_3 = \frac{z_c + cd}{k}. \tag{2-1}$$

The orthogonal matrix of rotation [rotf] from the XYZ coordinate system to the X'Y'Z' system is, therefore,

$$\begin{bmatrix} l_1 & m_1 & a \\ l_2 & m_2 & b \\ l_3 & m_3 & c \end{bmatrix}$$

where

$$\begin{aligned} m_1 &= l_3 b - l_2 c \\ m_2 &= l_1 c - l_3 a \\ m_3 &= l_2 a - l_1 b. \end{aligned} \tag{2-2}$$

The equation of the fracture plane reduces to $z' + d = 0$; the points (-ad, -bd, -cd) and (x_c, y_c, z_c) become (0, 0, -d) and (k, 0, -d), respectively (see Figure 2-2b).

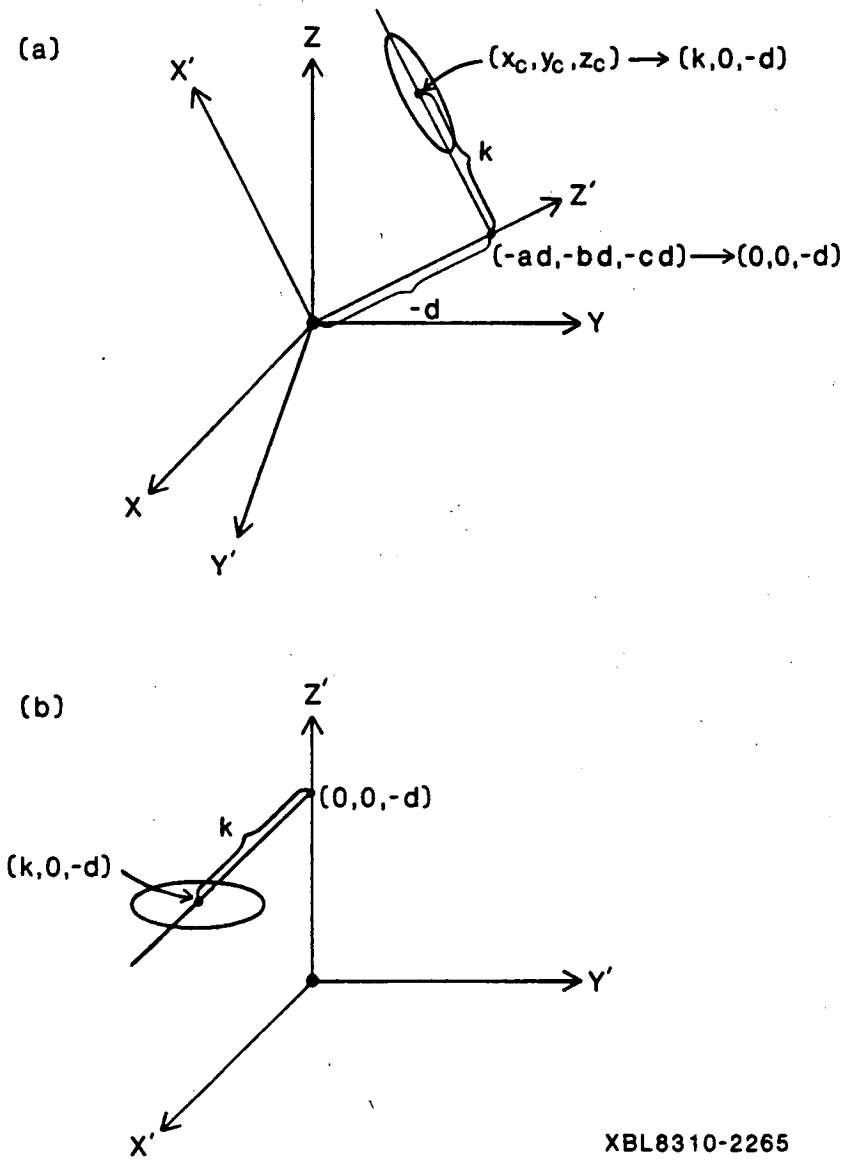


Figure 2-2. Rotation of coordinates to a simple coordinate system.

If the distance k equals zero, that is (x_c, y_c, z_c) and (-ad, -bc, -cd) are coincident, then the rotation matrix must be generated in a different manner. The Z-axis is rotated to have direction cosines a, b, and c, as before, but any other rotation is arbitrary. For convenience, the orientation angles of the fracture plane, ϕ' and θ', are used. By letting

$$l_1 = \cos\phi' \cos\theta'$$
$$l_2 = -\cos\phi' \sin\theta'$$
$$l_3 = -\sin\phi'$$

and

$$m_1 = \sin\theta'$$
$$m_2 = \cos\theta'$$
$$m_3 = 0,$$

an orthogonal rotation is assured. The equation of the fracture plane reduces to z' + d = 0 and point (x_c, y_c, z_c) becomes (0, 0, -d).

2.2.2 Channel Characteristics

Channel Centers.

The channel centers can be set to be randomly distributed or form a grid within the disc. The channel centers are generated by subroutine RANDXY to be randomly distributed throughout the disc. Coordinates of channel centers (x_c,y_c) are computed by generating pairs of random numbers, uniformly distributed between zero and one, then scaling then by multiplying by the diameter of the disc [diam], and subtracting the radius [rad]. This makes the center of the disc the origin of the rectangular coordinate system. The resultant pairs are rejected if they do not satisfy the condition $x_c^2 + y_c^2 \leq rad^2$, i.e., the point must be within the disc. The channel centers are generated by subroutine GRIDXY to form a grid within the disc. Coordinates of channel centers (x_c,y_c) are computed at even intervals along the x- or y-axis, depending on the number of channels and the specified orientation of the channels.

Orientations.

In order to specify the orientation characteristics of a given set of channels, the user inputs

both its average plunge direction $\bar{\beta}$ in the absolute three-dimensional system of coordinates, and a parameter σ describing the spread of the orientation distribution in each fracture disc. This parameter is generally the standard-deviation of the orientations, except in the case of a uniform distribution for which a range is given. A constant orientation must be input if a grid of channels is specified.

In the program, the orientation of each channel is recorded by the angle α between the channel line and the X' axis in the rotated system of coordinates local to each fracture. Both angles α and β are unique ways to define the orientation of a given channel on a disc. The relationship between these angles is shown by Figure 2.3. In order to generate the local orientation angles α for each channel on a given fracture disc with orientation angles ϕ and θ, the program first needs to determine which local mean orientation $\bar{\alpha}$ it should use. This local orientation $\bar{\alpha}$ must correspond to the average plunge direction $\bar{\beta}$ specified by the user. $\bar{\alpha}$ and σ are then used to generate the local orientations on the given fracture disc.

The plunge direction $\bar{\beta}$ corresponds on the fracture disc to a line in the direction of the vector $\vec{V}$ defined by the following global coordinates:

$$V_x = -\cos\bar{\beta}$$
$$V_y = \sin\bar{\beta}$$
$$V_z = \tan\phi\cos(\theta - \bar{\beta})$$

In the special case of a vertical fracture disc the angle $\bar{\beta}$ input by the user is interpreted as the inclination of the mean channel orientation with regard to the horizontal. In that case the vector $\vec{V}$ is then defined in global coordinates by

$$V_x = \cos\theta\cos\bar{\beta}$$
$$V_y = \sin\theta\cos\bar{\beta}$$
$$V_z = \sin\bar{\beta}$$

Using the rotation matrix [rotf] computed by subroutine ROFRAC, the coordinates of $\vec{V}$ in the local system, $V_{x'}$, $V_{y'}$ and $V_{z'}$ are computed. Then the angle $\bar{\alpha}$ is simply given by:

$$\bar{\alpha} = \arctan\left[\frac{V_{y'}}{V_{x'}}\right]$$

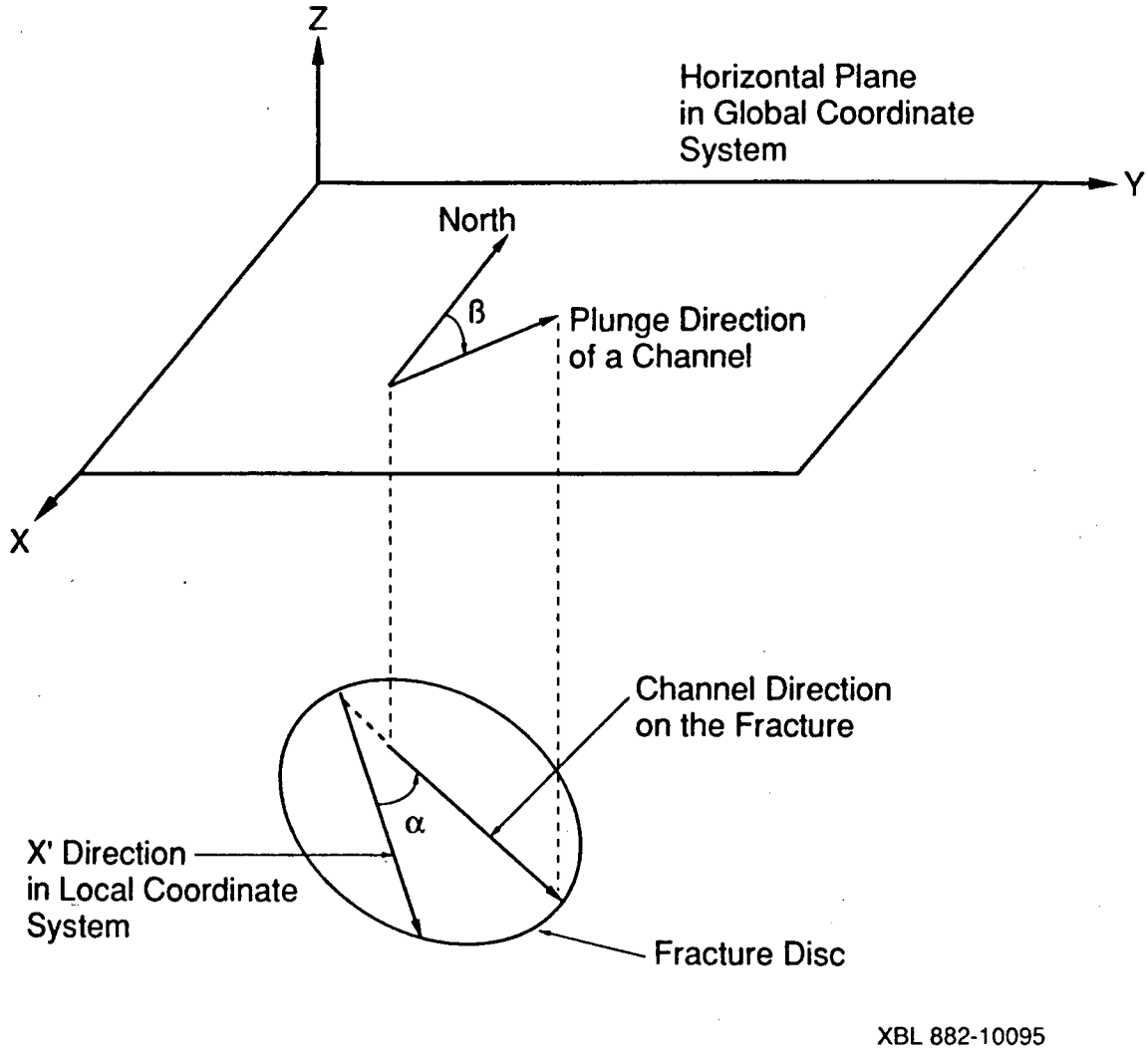


Figure 2-3. Angles α and β used to define the orientation of a channel.

If $V_{x'} = 0$, then $\bar{\alpha}$ is 90°.

Channel Lengths and Transmissivities.

The length l and transmissivity t of each channel are generated according to a normal, log-normal, exponential, uniform or constant distribution. The generation procedure for the first two distributions requires the mean [ev] and standard deviation [sd] for each channel set. The exponential distribution requires only the mean. The uniform distribution requires the mean and half range.

Transmissivities may be correlated with channel lengths. A global mean transmissivity may be specified, or the mean transmissivity in a given fracture disc may be taken proportional to the transmissivity generated by FMG3D for the fracture disc.

2.2.3 Statistical Simulation

Random Number Generator.

The statistical distribution subroutines and RANDXY use a random number generator called GGUBFS which is an International Mathematical and Statistical Library (IMSL) subroutine. GGUBFS returns random numbers uniformly distributed between zero and one and requires a double precision seed value [dseed]. GGUBFS returns a different random number each time it is called within a program. However, the same sequence of random numbers is produced each time the program is run with the same initial seed. This mode of operation is optionally overridden by generating an arbitrary seed,

$$\text{dseed} = \text{SECNDS}(0.0) * 100.0. \qquad (2\text{-}5)$$

SECNDS is a FORTRAN function subprogram which returns the system time of day in seconds less the value of its argument. An input flag [iranf] controls whether the seed is read or generated, and the initial seed is printed out. Since the seed defines the starting location for the random number generator, the user can reproduce a series of random numbers, i.e., reproduce a random channel system by inputting the same initial seed in a later run.

Random Generation of Channel Centers.

Subroutine RANDXY calls GGUBFS once for each coordinate of the channel center. The coordinates (x_c, y_c) are computed from the random numbers by the same equation

$$x_c \text{ (or } y_c) = \text{FLOAT [INT(2*rad*10}^n\text{*GGUBFS(d))]/10}^n - \text{rad,} \tag{2-6}$$

where n is the number of decimal places in the coordinate [itole],r is the radius of the fracture disc [rad],d is a dummy variable initially equal to the input or generated seed [dseed] then reset by GGUBFS. INT and FLOAT are intrinsic library functions which convert a real number to an integer by truncation and an integer to a real number, respectively. Truncating coordinates to n decimal places limits the minimum distance between channel centers.

2.2.4 Statistical Distributions Available

Normal Distribution.

In subroutine NORMAD, the sum S_N is calculated by calling GGUBFS twenty-five times and accumulating the sum,

$$S_N = \sum_{n=1}^{25} r_n \tag{2-7}$$

where r_n equals the value returned by a call to GGUBFS, r_n = GGUBFS(d). As shown by Hammersly and Hanscomb (1964), S_N is distributed normally with an expected value of 25/2 and a variance of 25/12; therefore,

$$S_N^* = \frac{\left[S_N - \frac{25}{2}\right]}{\sqrt{\frac{25}{12}}} \tag{2-8}$$

is distributed normally with expected value 0 and variance 1. If μ and σ are the expected value and standard deviation supplied by the user [ev and sd] then

$$x = \sigma S_N^* + \mu \tag{2-9}$$

is distributed normally, N (μ , σ) with the specified parameters. (Note that in this equation, x does not refer to a point coordinate.)

Lognormal Distribution.

If S_N^* and x are defined as in the previous section, then S_N^* is distributed N(0,1), and x is distributed N (μ , σ), and y = exp(x) is distributed lognormally. In terms of the parameters μ and σ of the normal distribution for x, the mean α and variance β^2 of the lognormal distribution are

$$\alpha = \exp(\mu)\exp(\sigma^2/2) \tag{2-10}$$

and

$$\beta^2 = \exp(\sigma^2+2\mu)[\exp(\sigma^2) - 1]. \tag{2-11}$$

Since the user will specify α and β, it is necessary to solve for μ and σ in terms of these variables:

$$\mu = 2 \ln \alpha - \frac{1}{2} \ln (\beta^2 + \alpha^2), \tag{2-12}$$

$$\sigma = \sqrt{\ln (\beta^2 + \alpha^2) - 2 \ln \alpha}.$$

Therefore, subroutine LOGNOD can calculate y from

$$y = \exp(\sigma S_N^* + \mu), \tag{2-13}$$

where μ and σ are defined above and α and β are specified by the user [ev and sd].

Exponential Distribution.

In subroutine EXPOND, μ is the expected value given by the user [ev] and

$$x = \mu \ln(1 - r) \tag{2-14}$$

is distributed exponentially, where r = GGUBFS(d).

Normal Distribution Correlating Two Variables.

Subroutine NORMD1 generates random variables, x, distributed normally where the expected value, μ, is proportional to another parameter, or to the logarithm of another parameter, x_1. This correlation may be used to compute channel aperture as a function of length. S_N^* is defined as before (Equation 2-8) and the standard deviation, σ, is supplied by the user [sd]. The user also supplies the y-intercept and the slope [ycept and slope] of a linear relationship between

mean values of the variable, x, and given values of the logarithm of x_1. The expected value of x, μ is computed,

$$\mu = \text{ycept} + \text{slope}*\log_{10}(x_1) \tag{2-15}$$

and x is computed as before, (Equation 2-9). Since this can result in values of x that are less than or equal to zero, which is not reasonable when x is the aperture, a minimum value for x is set in the subroutine.

Uniform Distribution.

Subroutine UNIFOD generates random variables, a_i, distributed uniformly over a given range, amin to amax:

$$a_i = (\text{amax} - \text{amin})*\text{GGUBFS}(\text{dseed}) + \text{amin} \tag{2-16}$$

Additional distributions.

The channel generation code can easily be modified to include additional distribution functions that are found to be appropriate for any of the channel characteristics.

2.3 Channel Endpoints

The coordinates of the endpoints of a channel are computed by subroutine ENDPTS from its orientation angle α, its length l, and the coordinates of its center. We first compute the quantities:

$$\begin{aligned} a &= -\sin(\alpha), \\ b &= \cos(\alpha), \\ dx &= l*b/2 \\ dy &= l*a/2, \end{aligned} \tag{2-17}$$

and then the endpoints (x_1, y_1) and (x_2, y_2) are given by:

$$\begin{aligned} x_1 &= x_c - dx \\ y_1 &= y_c - dy \\ x_2 &= x_c + dx \\ y_2 &= y_c + dy \end{aligned}$$

The quantities α, l, t, x_1, y_1, x_2, and y_2 are stored in the array [chan(maxcha,10)]. For each channel, the number of the fracture disc which supports it is stored in array [ifrach(maxcha)].

3.0 CHANNEL SYSTEM IN THE FLOW REGION

3.1 Flow Region

Channels are generated on every fracture generated by FMG3D. But in fact, we want to analyze the flow only in the channels which are in a given region of space that we call the "flow region". Boundary conditions will be applied at the boundaries of this flow region. The flow region is defined by one or more rectangular parallelepipeds (Figure 3-1). Flow will be computed in regions of space which are inside the first parallelepiped (the "proper" flow region) and outside the other ones (the "holes"). In this way, regions of simple geometry with boreholes or drifts can be represented.

In FMG3D, after all the fracture discs have been generated, the fracture discs lying entirely outside the flow region are discarded, and for the ones truncated by a flow region side, the intersections between them and flow region boundary planes are identified. These line segments are stored as boundary segments. Then, internal intersections between the remaining fracture discs are computed and truncated when they cross the boundary segments. Once this process is completed, a 3-D disc mesh has been fully specified. These operations are described in detail in the FMG3D report (Gilmour et al., 1986). Since all the flow region specifications are used by FMG3D, they are read by this program and transmitted directly between FMG3D and CHANGE. Therefore, there need be no specification for flow regions in the input deck written by the user for CHANGE.

Information about the disc mesh in the flow region is read by subroutine RINTER. This includes discs characteristics, disc-boundary intersections, disc-disc intersections. Then using this information and the channel characteristics generated previously, subroutine LIMIT determines the channels included in the flow region, discards channels totally outside the flow region, truncates channels lying partly outside the flow region, and finds which channel endpoints are on the boundaries of the flow region (Figure 3-2).

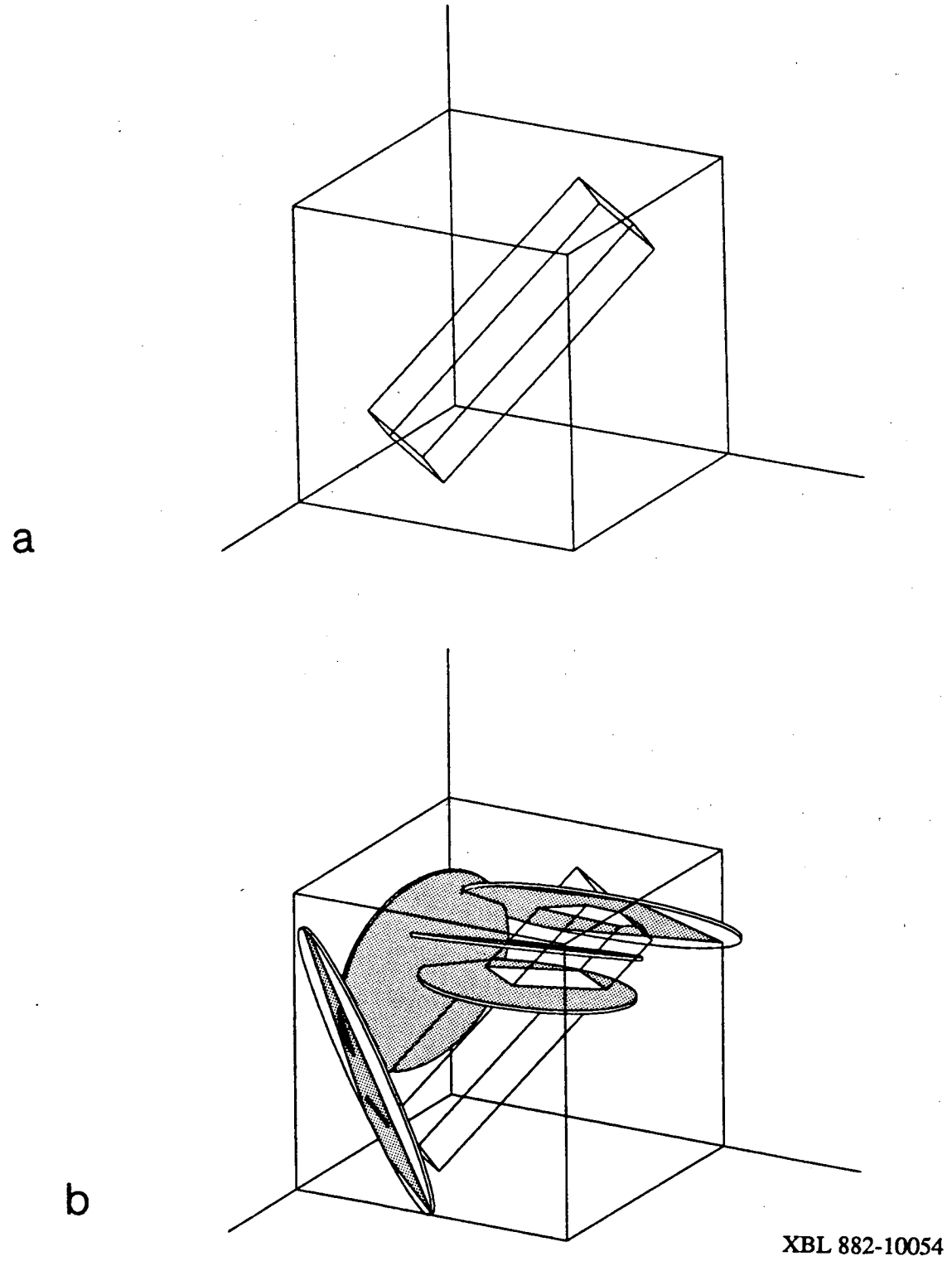


Figure 3-1. A proper flow region (cube) with a "hole" (long parallelepiped)
(a) Boundary, (b) Flow system shown as shaded area.

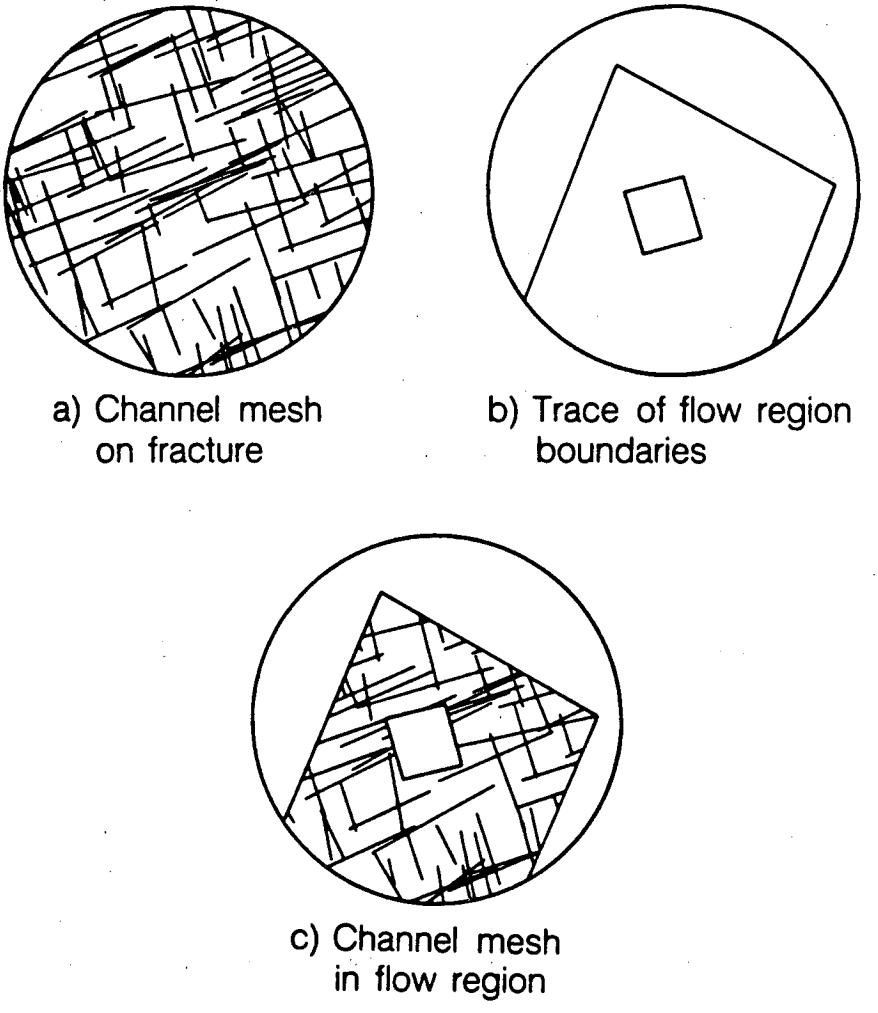


Figure 3-2. Limiting the mesh at flow region boundaries. (a) Channel mesh on fracture, (b) Trace of flow region boundaries, and (c) Channel mesh in flow region.

3.2 Channels in Flow Region

All the fracture discs generated by FMG3D are considered. If a fracture disc has no intersection with the flow region, it is simply skipped and the next fracture disc is considered. If at least part of the fracture is within the flow region, the rotation matrix [rotf] needed to go from the global system of coordinates to the local system of coordinates of the fracture (and vice-versa) is computed by subroutine ROFRAC as outlined in Chapter 2. Then, if the fracture has some boundary segments, these intersections are rotated from global to local coordinates by subroutine RFRCGLO using matrix [rotf].

We have now in each fracture a strictly 2-dimensional problem. All the channels in the fracture disc are considered in turn, and truncated if needed at the perimeter of the disc or at the boundary segments existing on the disc. First a channel is truncated at the perimeter of the disc on which it lies. Then it is truncated at the boundaries of the flow region. The flow region on the disc may be of any shape, convex or concave, and may be even made up of several distinct or overlapping regions. A channel may therefore be cut into two or more segments separated by "holes".

The endpoints of the channel and the intersections of the line supporting the channel, or "channel line", with the boundaries of the flow region divide this channel line into several segments. One or more of these segments may be part of the channel network inside the flow region, while other segments are outside the flow region or not between the channel endpoints. The points defining these segments are called "significant points". All the significant points along the channel line are recorded. Then these points are ordered, in order to find the segments they define along the channel line. Finally, each one of these segments lying between the endpoints of the channel is considered. A segment is kept if it is inside the flow region with regard to each parallelepiped that defines it. A segment is discarded if it is outside the flow region with regard to any one of the parallelepipeds that define the flow region.

3.2.1 Truncation of the Channel at the Disc Boundary.

The channel is truncated at the boundary of the disc on which it lies, and its endpoints and length are modified accordingly if necessary. The channel has endpoints coordinates X_1, Y_1, X_2, Y_2 in the local coordinate system and a length L [alen], and it lies on a fracture with radius R [rad] and center (0,0). A relative abscissa t_c on the channel line is defined by

$t_c = 0$ at the first endpoint M1 of the channel,

$t_c = 1$ at the second endpoint M2 of the channel.

When comparing two points M and M' on the channel line with relative abscissae t_c and t_c', we will say that M is "before" M' if $t_c < t_c'$.

Let x and y be the coordinates of an intersection point between the channel line and the disc boundary, then the following equations hold:

$$x = X_1 + t_c(X_2 - X_1) \tag{3-1}$$

$$y = Y_1 + t_c(Y_2 - Y_1)$$

$$x^2 + y^2 - R^2 = 0$$

Solving for t_c,

$$t_c^2[(X_2 - X_1)^2 + (Y_2 - Y_1)^2] + 2t_c[X_1(X_2 - X_1) + Y_1(Y_2 - Y_1)] + X_1^2 + Y_1^2 - R^2 = 0 \tag{3-2}$$

This second order equation is solved for its two roots t_c1 and t_c2. Note that since the channels are generated inside the disc,

1. there always should be two real roots to the equation (i.e. two intersections between the channel line and the disc boundary);
2. the smaller root t_c1 should be less than 1; and
3. the larger root t_c2 should be larger than 0.

If t_c1 is greater than zero, then the first endpoint is truncated:

$$X_1 \text{ is reset to } X_1 + t_c1 * (X_2 - X_1)$$

$$Y_1 \text{ is reset to } Y_1 + t_c1 * (Y_2 - Y_1)$$

$$t_c2 \text{ is reset to } (t_c2 - t_c1)/(1. - t_c1)$$

$$L \text{ is reset to } L * (1. - t_c1)$$

If t_c2 is less than one, then the second endpoint is truncated:

$$X_2 \text{ is reset to } X_1 + t_c2 * (X_2 - X_1)$$

$$Y_2 \text{ is reset to } Y_1 + t_c2 * (Y_2 - Y_1)$$

$$L \text{ is reset to } L * t_c2$$

3.2.2 Intersections between the Channel Line and the Disc Boundary Segments.

If the disc intersects the boundaries of the flow region, then the channels may be truncated at the boundary segments, or even be entirely outside. The program loops over the boundary segments on the disc in order to record all significant points on the channel line.

For each parallelepiped defining the flow region, either the proper flow region or any "hole", there may be zero, one, or two intersections with the channel line. A number is given to each group of points corresponding to a parallelepiped.

- the group "intersections with the proper flow region" has number one
- the groups "intersections with a hole" are given successive numbers starting at two, up to the number of holes plus one.

Note that a group of points always corresponds to a parallelepiped. However, if on a fracture disc there is no boundary segment corresponding to a given parallelepiped, there will be no point from the corresponding group on all the channel lines lying on this disc.

For each of the points in these groups, the program can find whether

1. the channel line is outside the flow region before the intersection and may be inside it after the intersection; or
2. the channel line may be inside the flow region before the intersection and is outside it after this intersection.

An "existence number" i_e is defined for each of the significant points. The absolute value of i_e is the number i of the group this point belongs to. Then i_e is set positive in case (1) above ("delete to exist" point), and negative in case (2) ("exist to delete" point).

A relative abscissa t_b is defined on each disc boundary segment on the fracture, in the same manner as for t_c. The abscissa t_b is 0 at the first endpoint of the boundary segment and 1 at its other endpoint.

For each significant point, the following information is recorded (Figure 3-3):

- relative abscissa t_c along the channel line,
- relative abscissa t_b along the disc boundary segment,
- number i_b of the boundary segment,
- existence number i_e as defined above.

Once all the boundary segments on the fracture have been considered, all the points recorded on the channel line are ordered according to their relative abscissa t_c.

3.2.3 Finding Existing Channel Segments.

As stated above, a segment on the channel line exists (i.e. is part of the channel network inside the flow region) if and only if 1) it lies between the endpoints of the channel and 2) it exists with regard to all the parallelepipeds defining the flow region. For each parallelepiped i, a flag Fl_i [keep(i)] is defined. For one given segment on the channel line, Fl_i is set to one if the segment is on the "inside" part of the channel line with regard to group i, and is zero otherwise. Once the Fl_i's for a segment have been set properly, in order to check if this segment exists, the program simply computes their product. The segment will correspond to an existing channel in the flow region if the product is nonzero for this segment.

For each channel, all the Fl_i flags are first initialized to properly reflect the position of the first endpoint M1 of the channel. This initialization sets them properly to determine if the first segment on the channel, starting from point M1, exists. Going from the first endpoint of the channel to the second, a loop over the points recorded on the line is executed. Any point number i_p

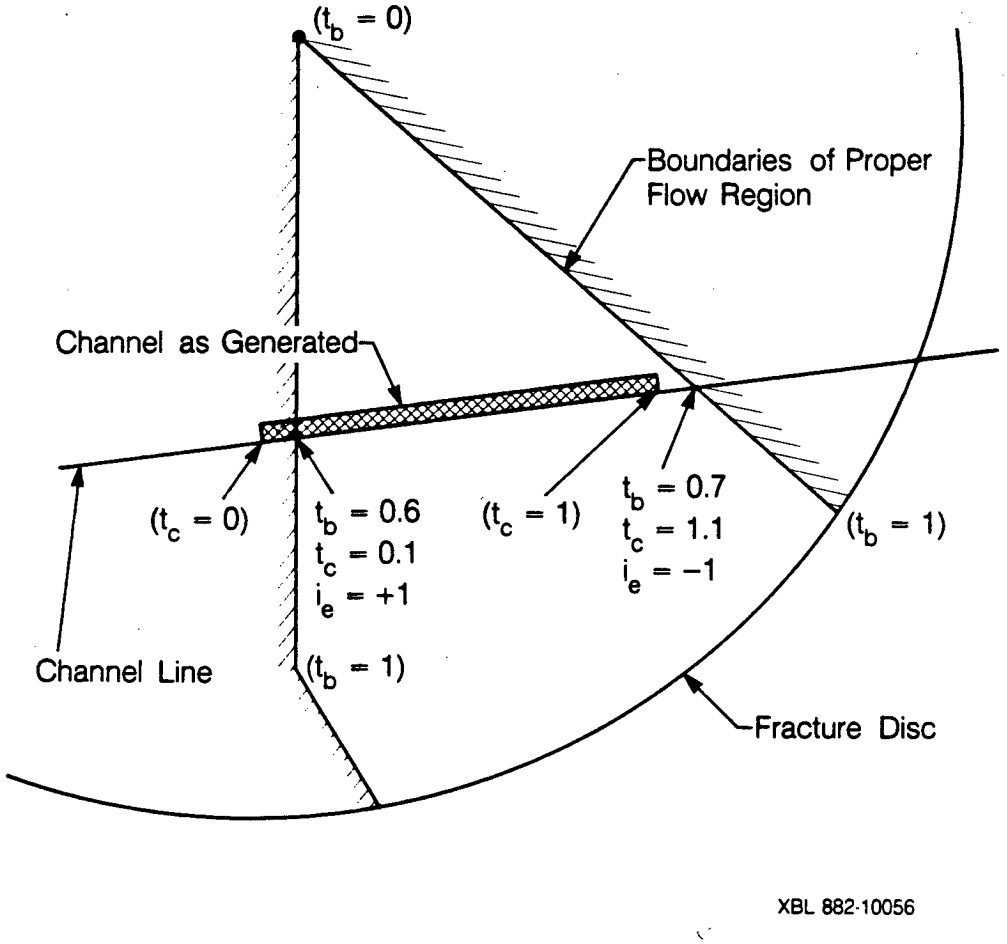


Figure 3-3. Information recorded along the channel line.

belongs to a given group i. The flag Fl_i is reset to one if i_e is positive and to zero if i_e is negative. The other flags Fl_j, $j \neq i$, are not changed. This sets all the flags properly for the segment s defined by points i_p and $i_p + 1$. The product of all the Fl_i's is then computed. The segment s exists if and only if this product is nonzero. In this case, this segment is added to the list of channels inside the flow region.

3.2.4 Recording Existing Channel Segments.

If we call t_c1 the value of t_c at point i_p, and t_c2 the value of t_c at point $i_p + 1$, then the length l and endpoint coordinates x_1, y_1, x_2, and y_2 of the new channel segment are:

$$l = L^*(t_c2 - t_c1),$$
$$x_1 = X_1 + t_c1(X_2 - X_{1)}$$
$$y_1 = Y_1 + t_c1(Y_2 - Y_1)$$
$$x_2 = X_1 + t_c2(X_2 - X_1)$$
$$y_2 = Y_1 + t_c2(Y_2 - Y_1)$$

When recording a new channel, an endpoint of the channel may be on a boundary segment. Also, the new channel may be overwritting the properties of a channel that has not yet been truncated. These two special cases must be taken care of.

If one or two endpoints of the channel are on a boundary segment, the channel must be recorded as a boundary channel. The number of the boundary segment and relative absissa t_b of the endpoint on the boundary segment are recorded.

Since we may be recording several new channels for one generated channel, the number of new channels created may at any time be larger than the number of the generated channel we are currently studying. Because we are using to store the properties of the new channels the same array [chan] we used for the channels at generation time, we may be overwriting on the properties of some generated channels we have not studied yet. Before this happens, subroutine MOVE is called to move the generated channels down the [chan] array. In order to prevent the calling of MOVE too early, a cushion is made when generating the channels by starting the channel numbers at 10*[nfrac], where [nfrac] is the number of fracture discs, instead of one. By starting the new channel numbers at one, we then allow 10*[nfrac]-1 extra channels to be recorded before

MOVE needs to be called. In fact, MOVE will be called only rarely, since in most cases the number of channels outside the flow region is much greater than the number of channels divided in two or more segments by the holes. Thus there are generally more channels deleted than created in subroutine LIMIT.

3.3 Recording Fracture Discs Intersections as Pipes.

The line mesh is constituted of both the channels generated by CHANGE and the fracture discs intersections computed by FMG3D. When referring to any element of the line mesh, we will use the term "pipe". The term "channel" refers exclusively to those pipes that were generated by CHANGE, and "fracture intersection" will be used for the pipes which are fracture-disc intersections.

Once all the channels on the fracture have been studied, the intersections of the fracture with other discs are considered. For each of them, the local coordinates of the intersection in this fracture are computed and recorded in array [inte4(8,maxint)]. If one of its endpoints is on the boundary of the flow region, then it is recorded as a boundary pipe.

4.0 PIPE SYSTEM TO BE USED IN FLOW MODEL

In order to calculate flow through the system, all pipe intersections must be located. Intersections between pipes and the boundaries of the flow region have already been determined. Therefore, the next step is to locate all intersections between pipes (internal nodes). This is done sequentially, starting with pipes intersecting the boundaries, then pipes intersecting these ones, and so on, until no new intersection is found. In this way, all the pipes from which there exists no path to any boundary, i.e. clusters isolated from the boundaries, are automatically discarded. This automatic discarding is useful when the channel network is used to compute steady-state flow with no matrix permeability. In this common case, the matrix representing the linear system to be solved is positive definite only if all isolated clusters have been removed. However, if for any special reason the user wants to keep these isolated clusters (for a pipe-matrix flow computation for example), a flag [ikeep] can be set in the input so that when all pipes connected to the boundaries have been delt with, isolated clusters are also considered.

4.1 Calculation of Pipe Intersections

The pipe mesh is built by subroutine INTERS from the boundaries to the inside of the flow region, level by level. Intersections of all the pipes intersecting the boundaries (pipes in level 1) with all the pipes, either in level 1 or not, are searched. All the pipes not in level one but which intersect a pipe in level one are put in level two, and are then screened for intersections with all other pipes, and so on.

When considering a pipe i and looking for its intersections with other pipes, only the pipes in the same(s) fracture disc(s) are screened. A pipe can be either a channel or a fracture disc intersection. For a channel, the pipes in one fracture only need to be screened. For a fracture disc intersection, the pipes in both fractures making up the intersection need to be screened. So the program first determines if a pipe is a channel or a fracture intersection and determines on which

fracture(s) pipes should be screened for intersections. On each of these fractures, channels are screened first and then fracture disc intersections. Only those which have not been considered yet are screened.

The computation of the intersection between two pipes i and j is performed in the plane of the fracture on which they are both laying. If a pipe is a fracture disc intersection, then among the two possible sets of local coordinates stored in array [inte4], the one corresponding to the fracture on which lays the other pipe is chosen. We now have (Figure 4-1) two line segments defined by their endpoints, M_{i1} (x_{i1}, y_{i1}) and M_{i2} (x_{i2}, y_{i2}) for pipe i, M_{j1} (x_{j1}, y_{j1}) and M_{j2} (x_{j2}, y_{j2}) for pipe j. Let us call M (x,y) their point of intersection, if it exists, and

$$x_i = x_{i1} - x_{i2}$$
$$y_i = y_{i1} - y_{i2}$$
$$x_j = x_{j1} - x_{j2}$$
$$y_j = y_{j1} - y_{j2}$$

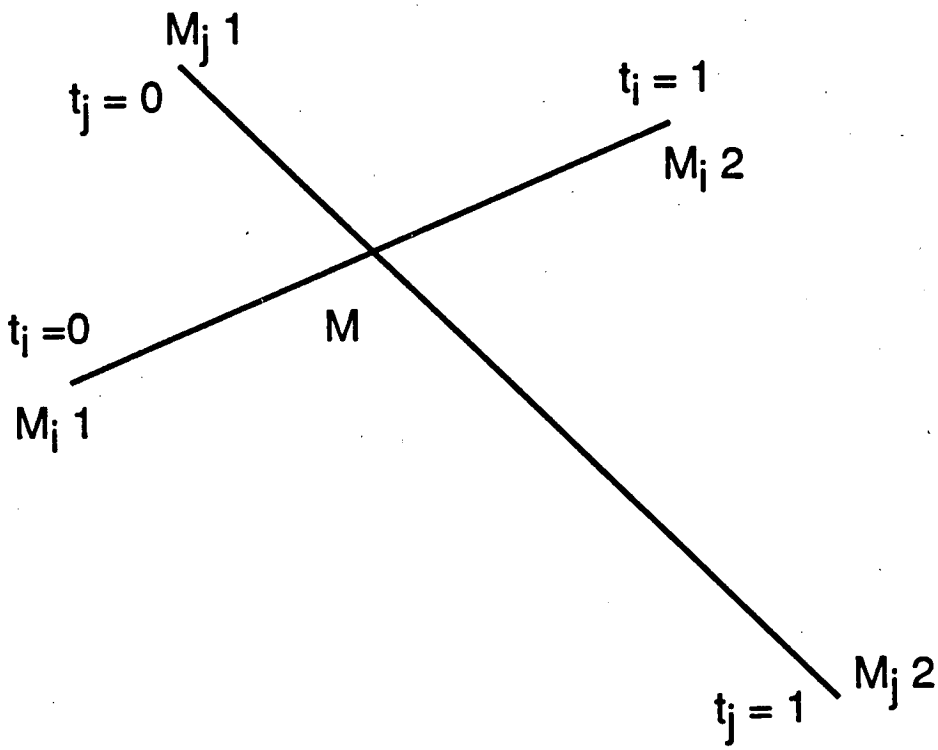
The two lines supporting the pipes will intersect between M_{i1} and M_{i2} if and only if the cross products

$$a_1 = \vec{M_{j2}M_{j1}} \times \vec{M_{j2}M_{i1}} = x_j(y_{i1} - y_{j2}) - y_j\,(x_{i1} - x_{j2}), \text{ and}$$
$$a_2 = \vec{M_{j2}M_{j1}} \times \vec{M_{j2}M_{i2}} = x_j(y_{i2} - y_{j2}) - y_j\,(x_{i2} - x_{j2})$$

have opposite signs. If $a_1 = a_2 = 0$ (in fact, if their absolute values are smaller than a given tolerance), the two pipes are on the same line. In this case, there will be an intersection if the two segments on the line overlap. If the two pipes are not on the same line, the relative abscissa t_j of the intersection along $\vec{M_{j1}M_{j2}}$ is computed using the fact that $\vec{M_{i1}M_{i2}}$ and [t_j $\vec{M_{j2}M_{j1}}$ - $\vec{M_{i2}M_{j1}}$] are parallel. Their cross product should therefore be zero, which after rearranging the equation yields:

$$t_j = y_i(x_{j1} - x_{i2}) - x_i(y_{j1} - y_{i2})/(a1 - a2)$$

This relative abscissa is checked for being between zero and one (i.e. intersection between M_{j1} and M_{j2}). If this is verified, then the two pipes intersect. The relative abscissa t_i along $\vec{M_{i1}M_{i2}}$ is computed using the same principle as for t_j. Vectors $\vec{M_{j1}M_{j2}}$ and [$\vec{M_{j2}M_{i1}}$ - t_i $\vec{M_{i2}M_{i1}}$] are



XBL 882-10057

Figure 4-1. Notations for pipes intersections.

parallel, and equaling their cross product to zero yields

$$t_i = \frac{a_1}{(a_1 - a_{2)}}$$

If the pipe j had not been encountered yet, its number is added to the list of pipes in next level. Also, since pipe i will not be screened when studying pipe j, the intersection on j is recorded.

After all the pipes in the fracture(s) on which pipe i is laying have been screened, the next pipe in the current level is considered. The process goes on until at the end of a level there is no pipe in the next level. At this point, if the flag [ikeep] has been set to two, indicating that isolated clusters should not be discarded, a pipe not yet encountered is put in the next level and the search for intersections resumes. If [ikeep] = 2, the search will end only when all pipes in the flow region have been considered.

4.2 Elimination of Nonconducting Pipes

In order for a pipe to conduct flow in steady state, it must contain at least two intersections, either with other pipes or with boundary intersections. Subroutine DISCHA determines which pipes have less than two intersections, eliminates them by adjusting the reference array [inext(maxcha)] and rearranges the intersection and boundary node arrays [iboun, jnod, knod, nodk, tboun, tnod]. The number of pipes [nchan], of pipes intersecting the boundaries [nbcha], of intersections [nnodes], and of boundary intersections [nbnd] are also reset . The elimination is iterative since the removal of one pipe may result in another pipe having less than two intersections. Elimination of non-conducting pipes is hydrologically correct for steady-state flow problems and often greatly reduces the size of the fluid flow analysis.

5.0 FINITE ELEMENT MESH

Boundary conditions, either in head or in flux, are read in for each parallelepiped and assigned to corresponding boundary nodes by subroutine BNDCON. Then the finite element mesh is output by subroutine WRENUM for input to program RENUM, and if the flag [iplot] has been set to 2 by the user, subroutine WLINES outputs the endpoints of the pipes for use by the plotting program DIMES.

5.1 Boundary Conditions

Boundary conditions are specified for each boundary parallelepiped. For each of them separately, a type of boundary conditions, and the parameters for this type of boundary conditions are input by the user. The three possible types of boundary conditions are: 1) linearly varying imposed heads producing a constant gradient throughout the flow region; 2) constant imposed head on each of the six sides; or 3) constant imposed flux through each of the six sides.

5.1.1 Constant Gradient Boundary Conditions

When using the chain of programs to measure directional permeabilities, flow region boundary conditions are set up to ensure a constant average gradient in the medium. Using a cubic flow region as an example, Figure 5-1 diagrams the head distribution on the six boundary planes. The input parameter for this type of boundary conditions is $\Delta\phi$, the difference in head between the inflow and outflow faces. The inflow face is assigned a head equal to $\Delta\phi$, and the outflow face is assigned a head of zero. The other four sides have fixed linearly distributed or wedge shaped head distributions with a value of $\Delta\phi$ along edges intersecting the inflow face and a value of 0 along the edges intersecting the outflow face. The head at any point on these four sides can, therefore, be found by linear interpolation. The direction of flow can be from side 1 to side 3, or from side 5 to side 6, or from side 2 to side 4, the side numbering convention being as shown in

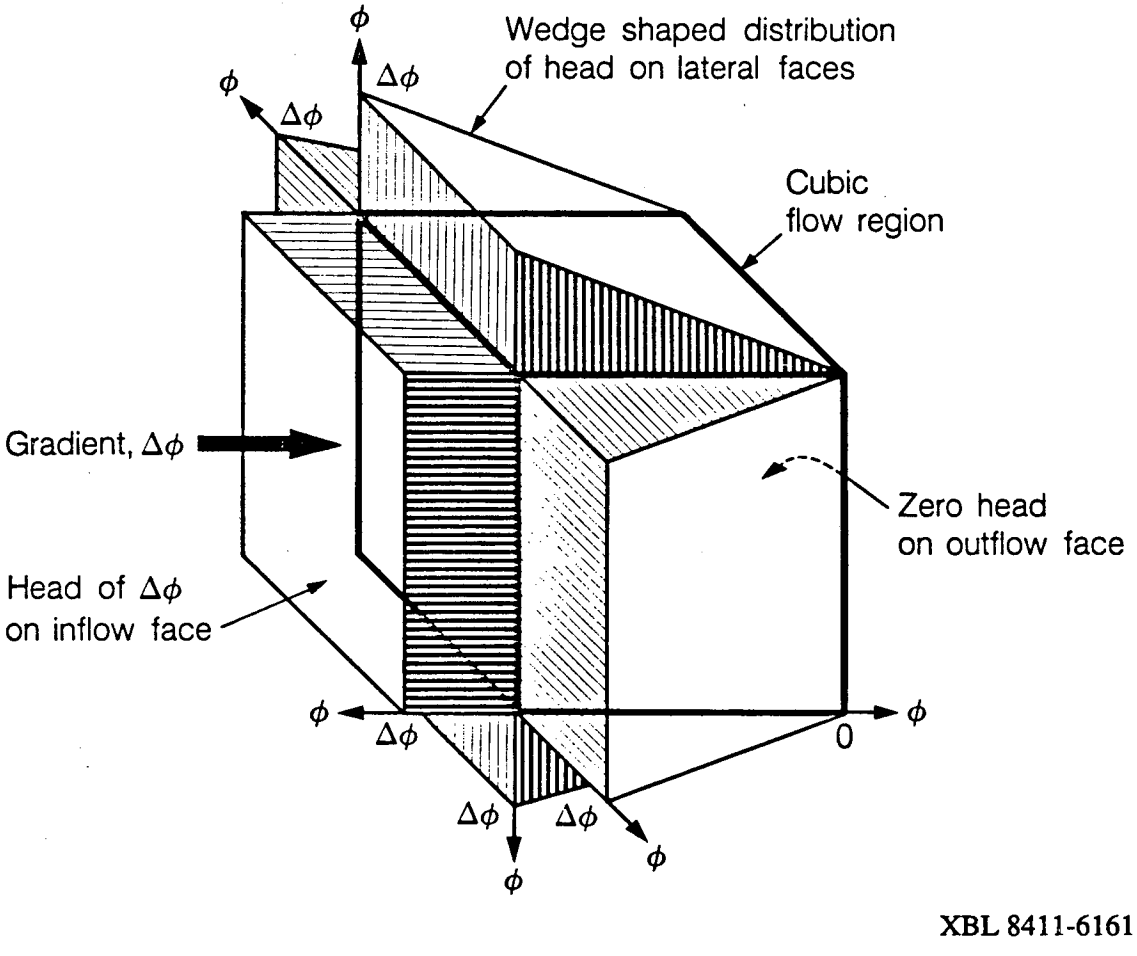


Figure 5-1. Boundary conditions used to produce a constant gradient.

Figure 5-2. If flow is to go in the opposite direction, from side 3 to side 1 for example, then the gradient, $\Delta\phi$, is negative. Note that the gradient will be constant only if there is no "hole" in the flow region. If "holes" with specific boundary conditions are introduced in the flow region, these holes are likely to perturb the gradient around them.

5.1.2 Constant Head or Constant Flux Boundary Conditions

Alternative types of boundary conditions may be used to solve problems other than directional permeability. In the case "constant imposed head", each boundary node is assigned a fixed head value according to the boundary plane in which it lies, regardless of the position of the node in that plane. The input parameters are then the six heads on the six sides of the parallelepiped. In the case "constant imposed flux", CHANGE just assigns to each boundary node the imposed flux specified by the user. The next program in the chain, RENUM will then shrink all the constant flux nodes from one hole into one single node. The input parameter for this type of boundary conditions is the value of the total imposed flux in the hole.

Subroutine BNDCON reads boundary condition parameters and computes and stores head or flux values. For each parallelepiped, up to three, [maxd], sets of boundary conditions are read in, either flow directions and gradients, [ibcc=1, ndir(maxd), delphi(maxd)], fixed head values for the six flow region boundary planes, [ibcc=2, seth(6,maxd)] or the fixed flux value for the hole [ibcc=3, seth(1,maxd)]. Head or flux values for each boundary node under each set of flow conditions are computed and stored in a linear array, [hd(maxh)], which is keyed by node number and the number of boundary conditions used.

5.2 Finite Element Mesh

Subroutine WRENUM writes data files to be used as input files by the the next program in the chain (see Figure 1-4), the network optimization program RENUM. One file is written for each different set of flow regions, and they are named RENUM01.DAT, RENUM02.DAT, etc... For each set of flow regions, WRENUM writes the following data:

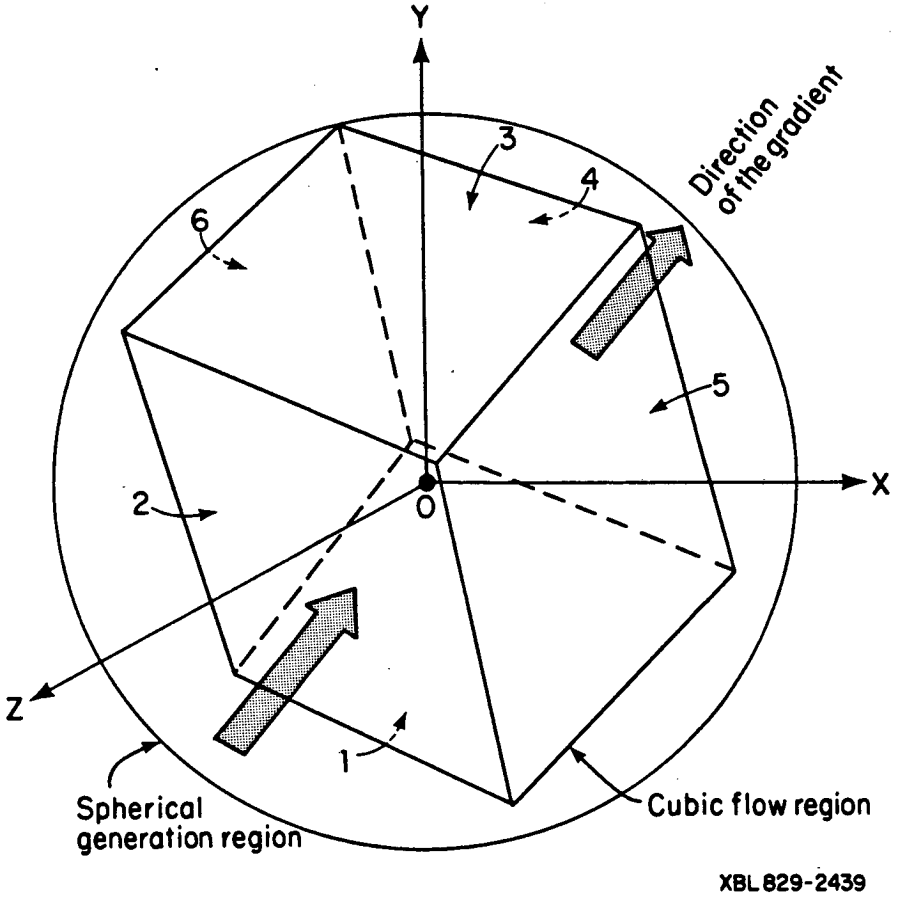


Figure 5-2. Side numbering convention for the flow region.

- run identification for both the CHANGE run and the FMG3D run that generated the data used for this run [iray,idate,oray,odate,title,title2]
- number of channels [nchan]
- all the global parameters defining the disc mesh:

  total number of fractures generated by FMG3D [nfracg]
  number of fractures in the flow region [nfrac]
  number of boundary fractures [nfracb]
  total number of fracture intersections [nbpt]
  starting location of internal intersections in the intersections arrays [ninst]
  minimum distance between two fracture centers [itole]
- radius of the generation region
- flow region parameters [xmesh, ymesh, zmesh, rophi, rothe]
- proper flow region boundary conditions parameters [ndir, delphi]
- physical constants [visc, spgr]
- number of elements [nelem]
- number of nodes [nnodes]
- number of boundary condition sets [maxd]
- value of the truncation flag [ikeep]
- value of the plotting flag [iplot]

The nodes and elements making up the line mesh are then output. Until this point, the position of a channel was recorded by the local coordinates of its endpoints in the plane of the fracture. For outputting the finite element mesh, we need to convert these local 2-D coordinates to global 3-D coordinates. This is done for each channel by getting the identification number of the fracture it is laying on, computing the rotation matrix [rotf(3,3)] associated with it using sub-routine ROFRAC, and then using [rotf] to transform the coordinates of the two endpoints. If, as happens very frequently, two channels studied consecutively are on the same fracture, then the call to ROFRAC is bypassed, and [rotf] is not recomputed. Note that this is done only for

channels, since we already know the 3-D coordinates of the fracture disc intersections.

The nodes are then output. First, a loop over boundary pipes is executed, and the boundary nodes on these boundary pipes are printed out, numbering them sequentially. Then if the user has specified that pipe endpoints should not be discarded (i.e. [ikeep]≥1) the pipe endpoint nodes are written. In this case, a second loop over the boundary pipes is performed in order to print the pipe endpoints on boundary pipes which are not on a boundary, and then a loop over the internal pipes prints their two endpoints. Once all boundary nodes and (if needed) pipe endpoints have been output, the internal nodes are printed by looping over node numbers and computing the coordinates of the node from the coordinates of the endpoints of the pipe and the relative coordinate of the node along the pipe.

Elements are output next. A loop over the pipes is executed. For each of them, all the nodes on the pipe are retrieved and sorted, and then the elements making up the pipe can be output. During the same loop, if [ikeep] is 0, the coordinates of each endpoint of a channel are set to the coordinates of the node closest to the endpoint. This will be useful when outputting the file for the graphic program DIMES.

5.3 Input to the Plotting Program

Subroutine WLINES writes files to be used as input files by the plotting program DIMES. One file is written for each different flow region, and they are named LINES01.DAT, LINES02.DAT, etc... A file contains only the endpoint coordinates of every pipe in the flow region. All the information about the disc mesh and the flow region or holes is contained in another file read by the plotting program, named DIMES01.DAT, DIMES02.DAT, etc.. and output by the fracture disc generation program FMG3D.

6.0 REFERENCES

Billaux, D. and J. E. Peterson, 1988. CHANGE: A Numerical Model for Three-Dimensional Modelling of Channelized Flow in Rock. User's Manual and Listing, Lawrence Berkeley Laboratory, Report No. 24911.

Billaux, D., S. Bodea and J. Long, 1988a. FMG, RENUM, LINEL, ELLFMG, ELLP and DIMES: Chain of Programs for Calculating and Analyzing Fluid Flow through Two-Dimensional Fracture Networks. Theory and Design, Lawrence Berkeley Laboratory, Report No. 24914.

Billaux, D., S. Bodea, J. E. Peterson and J. Long, 1988b. FMG, RENUM, LINEL, ELLFMG, ELLP and DIMES: Chain of Programs for Calculating and Analyzing Fluid Flow through Two-Dimensional Fracture Networks. User's Manuals and Listings, Lawrence Berkeley Laboratory, Report No. 24915.

Gentier, S., 1986. Morphologie et Comportement Hydromécanique d'une Fracture Naturelle Dans un Granite Sous Contrainte Normale. Doctoral thesis, Université d' Orléans, France, 640 pp.

Gilmour, N. M. P., D. Billaux and J. C. S. Long, 1986a. Models for Calculating Fluid Flow in Randomly Generated Three-Dimensional Networks of Disc Shaped Fractures. Theory and Design of FMG3D, DISCEL and DIMES. Lawrence Berkeley Laboratory Report No. 19515, 143 pp.

Gilmour, H. M. P., D. Billaux and J. C. S. Long, 1986b. Models for Calculating Fluid Flow in Randomly Generated Three-Dimensional Networks of Disc-Shaped Fractures. User's Manuals and Listings for FMG3D, DISCEL and DIMES. Lawrence Berkeley Laboratory, Report No 19516.

Karasaki, K., 1987. A New Advection - Dispersion Code for Calculating Transport in Fracture Networks, Lawrence Berkeley Laboratory, Earth Science Division 1986 Annual Report, LBL Report No. 22090.

Long, J. C. S., 1983. Investigation of Equivalent Porous Medium Permeability in Networks of Discontinuous Fractures, Ph.D. Dissertation, University of California, Berkeley, 277 pp.

Neretnieks, I., H. Abelin and L. Birgersson, 1987. Some Recent Observations of Channeling in Fractured Rocks - Its Potential Impact on Radionuclide Migration. DOE/AECL 1987 Conference on Geostatistical, Sensitivity, and Uncertainty Methods for Ground-Water Flow and Radionuclide Transport Modeling, San Francisco, 23 pp.

APPENDIX

A Numerical Model for 3-Dimensional Modelling
of Channelized Flow in Rocks

A NUMERICAL MODEL FOR 3-DIMENSIONAL MODELLING OF CHANNELIZED FLOW IN ROCKS

Daniel Billaux, Kenzi Karasaki and Jane C. S. Long
Lawrence Berkeley Laboratory
Berkeley, CA 94720, USA

Abstract: There is evidence that the voids in fractures often form tortuous channel networks. The channels can be considered as a mesh of quasi-one dimensional channels. In order to simulate such networks, we have designed and coded a new program, CHANGE (CHANnel GEnerator), which generates random channels on a given network of discs and outputs a three-dimensional finite-element grid of line elements. The channels in each fracture can be generated independently with random distributions of length , conductivity and orientation in the fracture plane. Boundary conditions are specified on the sides of the "flow region", and at the intersections of the channels with interior "holes" specified by the user to simulate boreholes. This code was used to generate an input deck for the transient flow code TRINET. A well-test with one well and four observation wells was simulated. The results show how connectivity plays a major role in governing the response of a channellized flow network.

Résumé: Les fractures forment parfois des chenaux tortueux. Ces chenaux peuvent être considérés comme un réseau de conduits quasiment monodimensionels. Pour simuler de tels réseaux, nous avons conçu et réalisé un nouveau programme, CHANGE (CHANnel GEnerator, ou générateur de chenaux), qui engendre des chenaux aléatoires sur un réseau donné de disques et produit un réseau tridimensionnel d'élément lignes. Les chenaux dans chaque fracture peuvent être engendrés indépendement selon des distributions aléatoires de longueur, de conductivité et d'orientation dans le plan de la fracture. Des conditions aux limites sont spécifiées sur les côtés de la region d'écoulement, et aux intersections des chenaux avec des "trous" spécifiés par l'utilisateur pour simuler des sondages par exemple. Ce programme a été utilisé pour réaliser un fichier d'entrée du programme de calcul d'écoulements transitoires TRINET. Un essai de pompage entre un sondage d'essai et quatre sondages d'observation a été simulé. Les résultats montrent le rôle majeur que joue la connectivité dans la réponse d'un réseau d'écoulement en chenaux.

Introduction

Two-dimensional and three-dimensional stochastic models of fracture flow have been built at LBL during the last five years. These models are based on the assumption that the flow in a given fracture can be approximated by parallel-plate flow. (Long, 1983; Gilmour et al., 1986) However, there is now evidence that the voids in fractures often form channels (Gentier, 1986; Neretnieks et al, 1987). In fact, deriving the characteristics of the channel system from the morphology of fracture voids obtained using a new casting technique is one the authors' current research topics (Gentier et al., 1988). These channels can be considered as a network of quasi-one dimensional channels. Building upon our previous experience with two and three-dimensional fracture networks, we have designed and coded a new program, CHANGE (CHANnel Generator), which generates random channels on a given network of discs and outputs a three-dimensional finite-element grid of line elements. This code was then used to generate an input for the transient flow code TRINET. A well-test with one input well and four observation wells was simulated. The results show how connectivity plays a major role in governing the response of a channellized flow network.

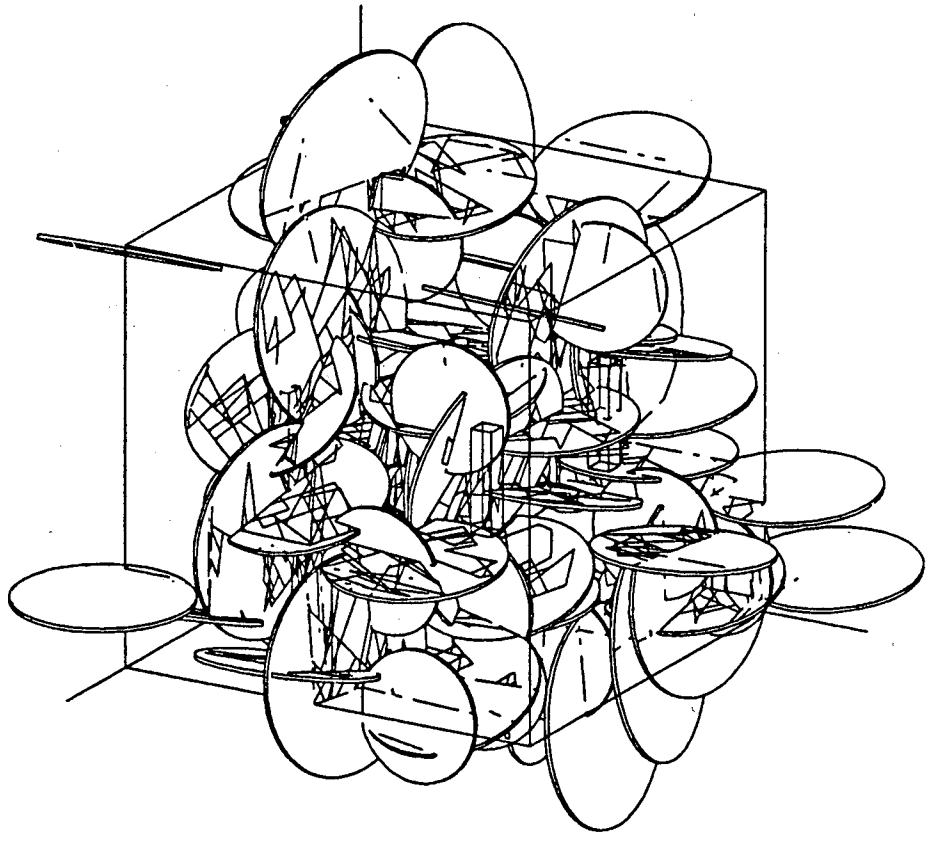
Channel generation and boundary conditions

Input to the program consists of the parameters necessary to specify a three-dimensional network of discs, the statistical properties needed for the channel network on the discs, and boundary conditions. The disc network is obtained by using the program FMG3D developed at LBL (Gilmour et al, 1986). The channels in each fracture disc can be generated independently with random distributions of length, conductivity and orientation in the fracture plane.

Complex distributions of channel characteristics can be obtained by superposing several channel sets on each fracture disc. If the fracture discs have been generated in several sets, then the characteristics of the channels can be controlled in each fracture set independently. In this way, different fracture morphologies can be reproduced in the same rock mass. Figure 1 shows part of a three-dimensional network of disc-shaped fractures with sub-networks of channels in the fractures. In Figure 2, only the channels network for the whole cubic region is plotted.

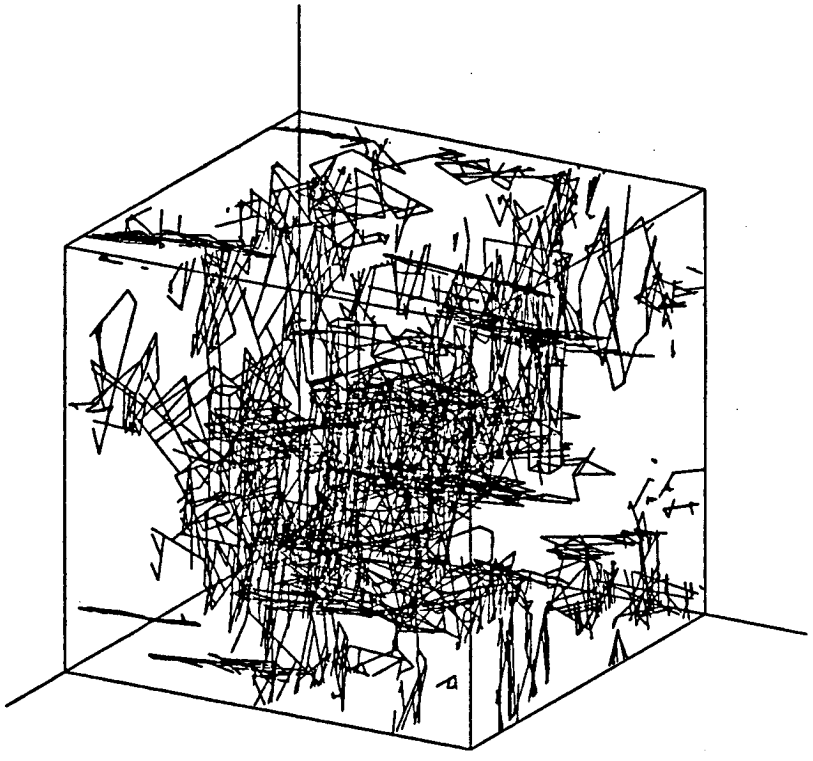
We currently generate channels over the whole area of each fracture disc, except when the discs lay partly outside the "flow region", i.e. outside of the total system in which flow occurs. However, the generation of channels within a fracture disc could be confined to any sub-area of the disc. This would effectively eliminate the constraint of having disc shaped fractures. In fact, the disc shape is only necessary in locating intersections between fractures.

The total flow region must be a rectangle parallelepiped with any orientation in space. In addition, any number of rectangular-shaped "holes" can be specified within the region in order to simulate sections of drifts or boreholes. Five such boreholes are present in the network shown on figures 1 and 2. In figure 3, most of the fractures have been removed, to reveal the locations of the "holes" used to simulate the open sections of boreholes.



- XBL 881-324 -

Figure 1 Part of a mesh of channels on discs. The broken lines are intersections with discs not shown.



- XBL 881-325 -

Figure 2 Channels only from mesh in Figure 1.

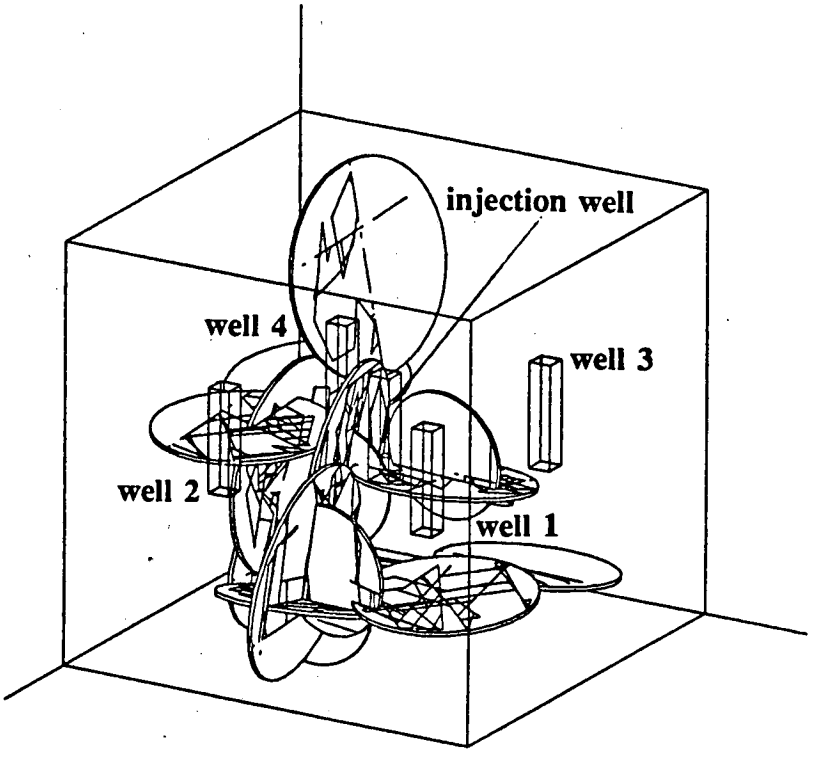


Figure 3 Modelling boreholes.

Flow mesh

The intersections between fracture discs are treated as a separate class of conductors in this model. As such, they can be characterized separately. Currently, the disc intersections are treated in the model as high conductivity channels. This seems to be geologically most reasonable. However, these intersections can be treated otherwise, if that is desirable. The union of the generated channels and the fracture disc intersections makes up the flow mesh and we use the generic term "pipe" for both kinds of conductors.

Before the mesh can be used to compute flow, the program must determine which channels are in the flow region, and which ones are truncated at the boundaries of the flow region. The boundary conditions will be applied to the endpoints of channels which are truncated. Once the mesh within the flow region has been identified, intersections between pipes must be found. This is done from the boundaries inward, determining all the intersections between boundary pipes and any other pipes, then intersections between these pipes and other ones, and so on. In this way, pipes not connected to a boundary are never included. A special option can disable this feature and effectively produce a mesh containing all the pipes in the flow region, for pipe-matrix flow studies for example. If a pipe is a channel within a fracture disc, only other pipes in the same disc can intersect it. If a pipe is a fracture disc intersection, it lies on two different discs, and the pipes on either disc forming the intersection must be searched for intersections with that particular pipe.

All the computations for truncating channels and finding intersections between pipes are performed in a local coordinate system defined by the plane of the relevant fracture. This saves both computer memory an processing time. Once all intersections have been found, the program can identify and eliminate simple dead-ends, i.e. pipe endpoints or pipes showing only one intersection with the rest of the mesh. This option can be overriden for transient flow computations. The program produces a mesh with pipe segments as line elements and pipe intersections as nodes and with specifications of head or flux imposed at the boundary nodes.

Computing flow

The mesh is then processed by program RENUM, which optimizes the finite element network by merging nodes, removing complex dead-ends, and then renumbering the nodes. The short-circuit effect of wells is taken into account by shrinking all the boundary nodes on a given imposed flux "hole" into one node. This is equivalent to assuming that the well has an infinite transmissivity. Nodes very close to each other are also merged. The program optionally removes complex dead-ends (Billaux and Fuller, 1988). The nodes are finally renumbered using an algorithm published by Cuthil and McKee (1969) in order to minimize the bandwidth of the corresponding linear system of equations.

Program TRINET (Karasaki, 1987) can then compute flow and transport in the network. TRINET incorporates the Lagrangian and Eulerian schemes with adaptive gridding to solve the advection-dispersion equation. The model avoids numerical dispersion by creating new Eulerian grid points to preserve the exact shape of the front

instead of interpolating the advected profile back to the original Eulerian grid. However, the mass transport option was not used in this study. The head distribution in the network is solved using the usual Galerkin finite element method with linear shape functions. The flow can be either steady-state or transient, where the time derivative is treated in a finite difference manner.

It is assumed that the flow through each pipe can be represented by flow in porous media. The authors believe that this assumption is to reality than the normal approach of a smooth wall pipe. Therefore, the governing equation for the transient flow of slightly compressible water through a pipe can be written as:

$$K\frac{\partial^2 h}{\partial x^2} = S_s\frac{\partial h}{\partial t}\,, \tag{1}$$

where h is the hydraulic head, x is along the direction of the pipe orientation (local coordinate system), T is the permeability of the media, and S_s is its specific storage. It is assumed that the permeability is constant and not a function of pressure and that the flow is laminar. These assumptions may not be valid under some circumstances, especially where the pressure in the pipes exceeds the overburden pressure so that the permeability of the pipe becomes dependent on pressure.

The finite element formulation is based on the Galerkin approach. The integration of basis functions is done over the length of a line element and the general form of the resulting element equations in matrix notation is:

$$\left[\frac{[A]^t}{\Delta t} + \theta\cdot[B]^t\right]\cdot\{h\}^{t+\Delta t} = \{F\} + \left[\frac{[A]^t}{\Delta t} - (1-\theta)\cdot[B]^t\right]\cdot\{h\}^t, \tag{2}$$

where [A] is the storage coefficient matrix, [B] is the permeability matrix, {F} contains boundary conditions, {h} is the head vector, and θ is the time weighting parameter. The above equations are solved simultaneously for the head at each node.

Well Test Simulation

In order to show how this new fracture model can help us, a double packer pressure injection test was simulated using the network and wells shown in figures 1, 2 and 3. This simulation was not carried out to reflect conditions at any real site, but rather to make clear how such "numerical experiments" can help us relate the flow properties of the medium to its geometric characteristics. The flow region is a 20 m by 20 m by 20 m cube, the wells are 1 m by 1 m by 5 m parallelepipeds. The injection well is located in the center of the flow region, and the four other wells are located at the corners of a 10 m by 10 m horizontal square. The network consists of 2907 channels and disc intersections, lying on 149 fracture discs generated at random in three sub-orthogonal sets. The resulting finite element mesh has 8534 nodes and 13215 line elements. A constant flow was imposed at the injection well. The four other wells were used as observation wells. Each of the five wells was also allowed to act as a short circuit between the fractures it intersects. The outer boundaries where given a constant head to simulate an open system.

Figure 4 shows the transient response at the four surrounding observation wells by comparison with the effects at the injection well. To help in the interpretation of the simulated well test, Figure 5 has been prepared to show how the pipes are

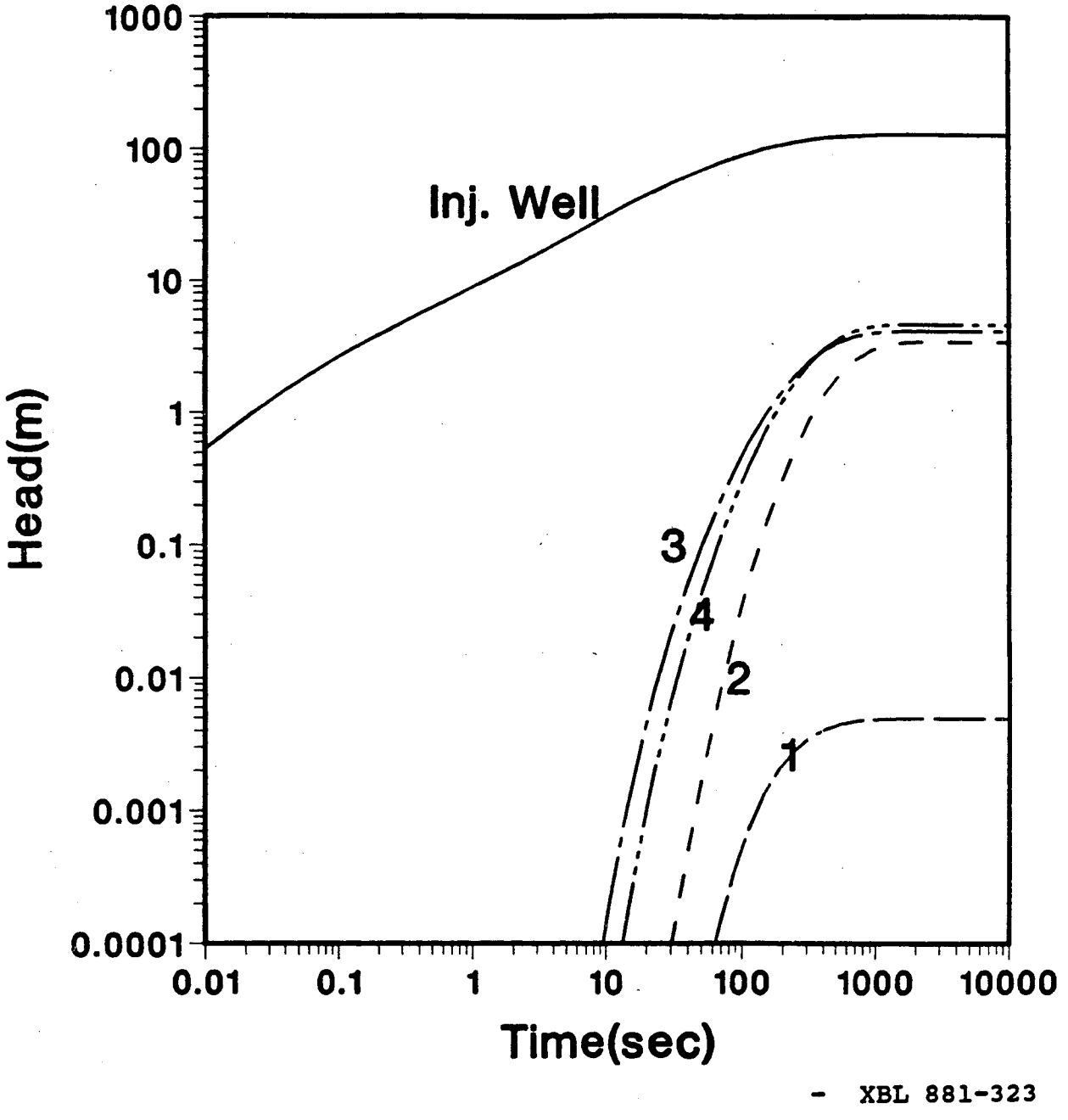


Figure 4 Head versus time at the injection well and the four observation boreholes.

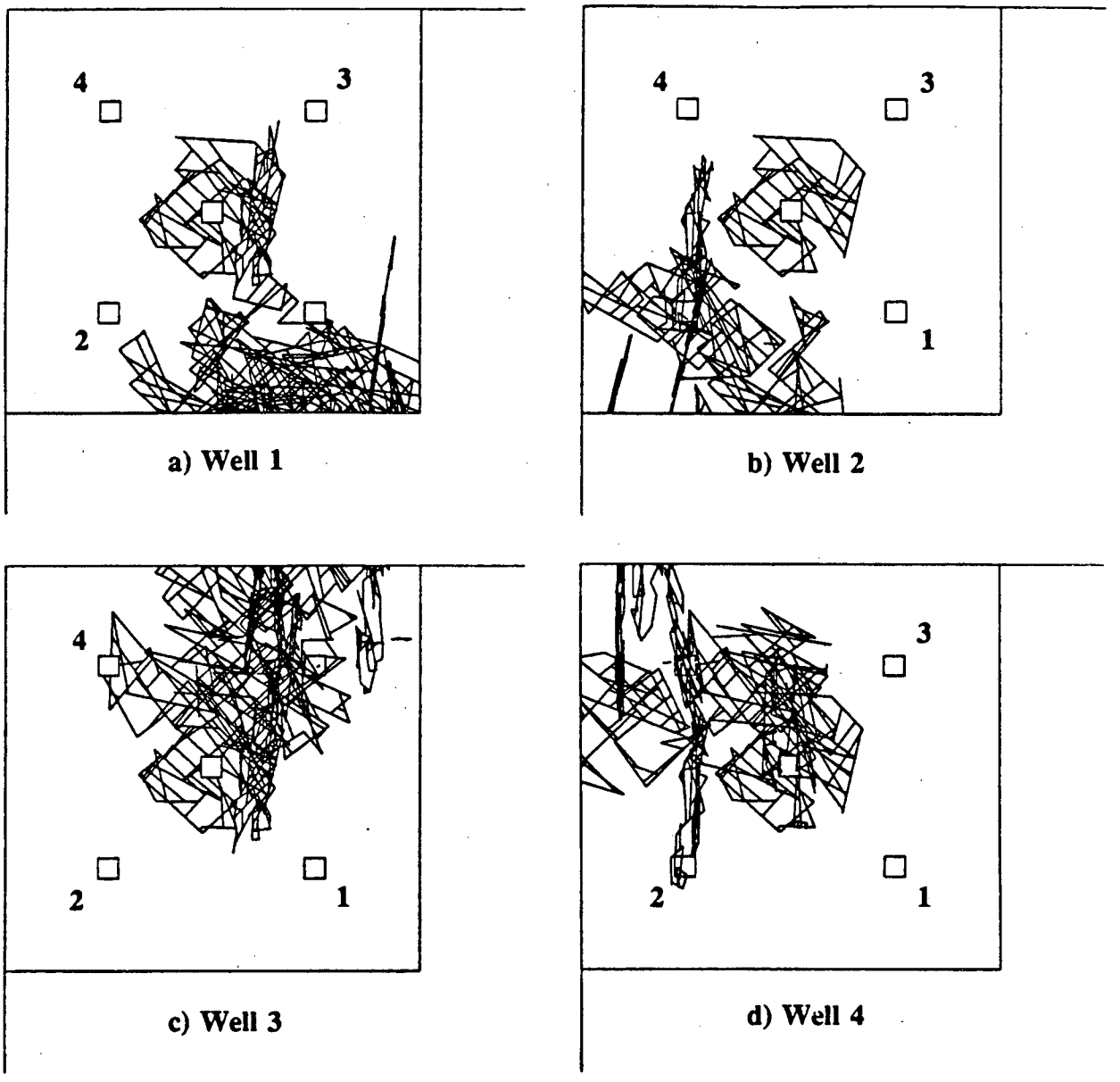


Figure 5 Channels surrounding each observation well.

clustered around each observation well. This was done by examining an arbitrary spherical region around each well that extended out to a radial distance of 4 m from the center of the well. All pipes within any fracture disc that intersects the spherical region were then plotted as projections on a horizontal plane. This produced a map which shows those pipes that have immediate pathways to each well. Note that on Figure 5-b, no connection can be seen between Well 2 and the injection well. This does not mean that the two wells are not connected, but simply that any path connecting them goes through a fracture outside the spherical region. Any path between Well 2 and the injection well must therefore be fairly long.

At large times all the curves flatten because of the constant head boundary condition imposed on all sides of the cubic flow region. In a three-dimensional regular lattice of conductors, or in a homogeneous isotropic porous medium, the curves for the four observation wells would be identical because the flow would be radially symetrical. However, as can be seen in figure 4, this is not the case.

Note that the transient effects take the longest time to reach Well 1 and at a much lower head value. This indicates that the effective hydraulic diffusivity to this well is the lowest of the four wells, and that there is a good hydraulic connection between Well 1 and the boundaries. These findings are also evident in figure 5-a. First, it can be seen that only two flow paths exist between Well 1 and the injection well. Secondly, Well 1 appears on the figure to be very well connected to the boundaries.

Another interesting observation can be made by comparing the transient and steady-state behaviors. One might conclude from the steady-state results that permeability in the direction of Well 4 is somewhat more than in the direction of Well 3. However, the transient responses of these two wells suggest an opposite effect; it can be seen that the hydraulic diffusivity of the system controlling flow to Well 3 is more effective than that to Well 4. This inversion of results can be explained by examining figure 5. One can see that Well 3 is better connected to the injection well than is Well 4, which explains the earlier response; and that Well 3 is also better connected to the outer boundaries, which explains the somewhat lower steady-state result.

It should also be noted on figure 5-d that Well 2 is well connected with Well 4, whereas from figure 5-b is is evident that the connections between Well 2 and the injection well must be relatively quite long. This is the reason for the lag in the transient response at Well 2 and the final steady-state response that is almost the same as that at Well 4.

Variations in head at the injection well provide another interesting result. Note the subtle but definite inflection in the curve on figure 4 at about 2 seconds. Up to this point, the pressure transients were propagating through a system close to the well, but after this time, propagation of the transient effects encountered a greater resistance to flow. This could be due to the flow reaching the limits of the fracture intersecting the injection well. The pressure transients can then propagate only through the intersections between this fracture and the rest of the system.

Conclusion

The newly completed chain of programs FMG3D, CHANGE, RENUM, and TRINET, is now a unique tool for the modelling of flow and transport in very complex channelized fractured rock geometries. A simple example of well-test simulation shows how such a tool can give us more insight into the relationship between the three-dimensional geometry of fractured rocks and their flow properties. The connectivity of the mesh is a major factor governing the response to the numerical injection test. This is likely to be also true in the field, and leads to fundamental differences with the behavior of a classical porous medium. Getting the right parameters to input in the numerical model is the emphasis in the next stage of this research. This involves the study of the morphology of the voids of fractures under stress in the laboratory and in the field.

References

Billaux (D.) and Fuller (P.) 1988.- An Algorithm for Mesh Simplification Applied to Fracture Hydrology.- In : Journal of the International Association for Mathematical Geology, New York (accepted for publication)

Cuthil (E.) and McKee (J.) 1969.- Reducing the Bandwidth of Sparse Symmetric Matrices.- In : Proceedings of the 24th Nat. Conf. Assoc. Comput. Mach., ACM Publications, New York, 69 p.

Gentier (S.) 1986.- Morphologie et Comportement Hydromécanique d'une Fracture Naturelle Dans un Granite Sous Contrainte Normale.- Doctoral thesis, Université d' Orléans, Orléans, France, 640 p.

Gentier (S), Billaux (D.), and van Vliet (L.) 1988.- Laboratory Testing of the Voids of a Fracture.- submitted to Rock Mechanics and Rock Engineering, Springer-Verlag, Wien.

Gilmour (P.), Billaux (D.) and Long (J. C. S.) 1986.- Models for Calculating Fluid Flow in Randomly Generated Three-Dimensional Networks of Disc-Shaped Fractures - Theory and Design of FMG3D, DISCEL, and DIMES.- Lawrence Berkeley Laboratory report number 19515, 143 p.

Karasaki (K.) 1987.- A New Advection-Dispersion Code for Calculating Transport in Fracture Networks.- In : Lawrence Berkeley Laboratory Earth Science Division 1986 Annual Report, LBL report number 22090, Berkeley, pp. 55-57.

Long (J. C. S.) 1983.- Investigation of Equivalent Porous Medium Permeability in Networks of Discontinuous Fractures.- Ph.D Dissertation, University of California, Berkeley, 277 p.

Neretnieks (I.), Abelin (H.) and Birgersson (L.) 1987.- Some Recent Observations of Channeling in Fractured Rocks - Its Potential Impact on Radionuclide Migration. - In : DOE/AECL 1987 Conference on Geostatistical, Sensitivity, and Uncertainty

Methods for Ground-Water Flow and Radionuclide Transport Modeling, San Francisco, 23 p.

LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
INFORMATION RESOURCES DEPARTMENT
BERKELEY, CALIFORNIA 94720