Title

CHANGE: A Numerical Model for Three-Dimensional Modelling of Channelized Flow in Rock. User's Manual and Listing

Authors

Billaux, D.
Peterson, J.E.

Publication Date

1990-03-01

LBL-24911
UC-403

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

EARTH SCIENCES DIVISION

CHANGE: A Numerical Model for Three-Dimensional Modelling of Channelized Flow in Rock. User's Manual and Listing.

D. Billaux and J.E. Peterson, Jr.

March 1990

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

DISCLAIMER

LBL-24911

CHANGE: A Numerical Model for Three-Dimensional Modelling of Channelized Flow in Rock. User's Manual and Listing

Daniel Billaux 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.

Table of Contents

1.0 INTRODUCTION

The purpose of this report is to provide the user with sufficient information to run the program CHANGE. A companion report explains the theory and design of the program (CHANGE: A Numerical Model for 3-D Modelling of Channelized Flow in Rock - Theory and Design). Using the two reports, a thorough understanding of the code is possible. However, since this program is part of a chain of programs (as described in the companion report), the user should know how to run program FMG3D, which must be run before CHANGE to provide the discs on which the channels will be generated, programs RENUM and LINEL or TRINET, which use the output from CHANGE and compute flow in the pipe mesh generated by CHANGE. Program DIMES may also be needed, if a plot of part or all of the fracture-channel mesh is necessary. Information on programs FMG3D and DIMES can be found in Gilmour et al. (1986a and b). Information on programs RENUM and LINEL can be found in Billaux et al. (1988a and b). Information on program TRINET can be found in Karasaki (1987).

This report should familiarize the user with program options and modes of operation, input variables, input and output files. In addition, information not strictly needed to run the program, but useful in understanding its internal structure and the flow of computation is provided in the appendices. The appendices cover program variables and arrays, subroutines outlines, a short description of each subroutine, and finally a listing of the code.

CHANGE is written in FORTRAN-77 and versions run SUN-4, VAX 11/86 and CONVEX C-1 computers. It is easily portable to any computer with a FORTRAN-77 compiler. In most cases, all that has to be changed is the OPEN and INCLUDE statements, which define input and output files and arrays. Large arrays are dimensioned by constants set in "parameter" statements. An "include" statement is used to insert the dimensioning constants and the COMMON blocks in each subroutine. The allowable problem size, controlled by the maximum number of fracture discs, of channels, of pipe intersections, etc., can be changed by simply editing the included parameter statement and then recompiling and linking the program. Preceding the FORTRAN listing is a list of the files making up the program. Also given is the "makefile" used to compile and link the program on a UNIX system.

2.0 PROGRAM CHANGE

Several options are available for output generated by program CHANGE. First the user selects one of three modes of operation, or ways in which the program can be run:

1. Generation of secondary channels on a primary fracture system.
2. Generation of secondary channels on a primary fracture system and determination of the conducting pipe system in one or more flow regions.
3. Determination of the conducting pipe system in one or more flow regions from the channels previously generated on a primary fracture system.

The mode of operation is controlled by the input variable icont, (the values of icont=1, 2, and 3, correspond to the option numbers above).

Table 2-1 lists the input variables and their formats by groups. The input groups required to run the program depend on the selected mode of operation. Table 2-2 shows which input groups are read for each value of control variable icont. Groups 1, 3, 4 and 7 are read from the file CHANGE.INP, which must be written by the user. Group 2 is read from the unformatted sequential file FRAC3D.DAT. This file is written by the program FMG3D when it has finished generating fractures, before it selects which fractures are in the flow region(s). Group 5 is read from file CHANGEnn.DAT. This formatted file is written by FMG3D after it has found all the intersections between fractures in the flow region. There is one file for each flow region, named sequentially starting with CHANGE01.DAT, then CHANGE02.DAT, etc.. Group 6 is read from the unformatted file CHAN.DAT. This data group is written if icont equals 1 or 2, but contains the fracture system from a previous run when icont equals 3. Data group 4 is repeated for each channel set. Data group 7 may be repeated for each parallelepiped in each flow region. If this data group is present only once and there are several flow regions, the same data is used for all different flow regions. Data Group 6 is read for each flow region when icont equals 2 or 3 to restore the primary channel system which is altered when channels are truncated or deleted. The input variables in file CHANGE.INP, groups 1, 3, 4 and 7 are described in Table 2-3. The other files contain only information passed directly from program to program, so the variables in these files are considered as internal variables and described in Appendix A.

Table 2-1. CHANGE Input Variables and Formats

Group	Variable	Format	Input Unit
1	icont,iplot,iunits	10x,i5,2(15x,i5)	1
	iranf,dseed	10x,i5,15x,d15.8	
	title(i),i=1,2	a	
	nchst	10x,i5	
2	oray,odate	unformatted	5
	title2(i),i=1,2	sequential file	
	nsets,itole,ikeep		
	nfrac		
	((frac(j,i),j=1,12),isetfr(i), i=1,nfrac)		
3	((isetch(iset,i),i=1,10),iset=1,nsets)	10x,10i5	1
4	icent,idens	2(10xi5)	1
	rlamb (if idens = 0)	10x,e10.4	
	or nchen (if idens=1)		
	ichar (for orientation)	10x,i5	
	const(if ichar=2)	10x,e10.4	
	or		
	idist,ev,sd(if ichar=3)	10x,i5,15x,f10.4,10x,f10.4	
	ichar (for length)	10x,i5	
	const (if ichar=2)	10x,e10.4	
	or		
	idist,ev,sd (if ichar=3 or 6)	10x,i5,15x,f10.4,10x,f10.4	
	ichar (for transmissivity)	10x,i5	
	const (if ichar=2)	10x,e10.4	
	or		
	idist,ev,sd (if ichar=3 or 6)	10x,i5,15x,f10.4,10x,f10.4	
	or		
	ycept,slope,sd (if ichar=4 or 5)	3(10x,f10.4)	
5	nfracg,nfrac,nfracb	10x,i5,15x,i5,15x,i5	9
	nbpt,inst,ninb	10x,i5,15x,i5,15x,i5	
	rgene,nhole,nbhol	10x,f10.0,10x,i5,10x,i5	
	xmesh,ymexh,zmesh	3(10x,f10.0)	
	rophi,rothe	2(10x,f10.0)	
	(frcoef(i,j),i=1,4),j=1,6*(nhole+1))	4(1x,e12.5)	

	iref(i),i=1,nfrac	15i5	
	((frac(j,i),j=1,12),i=1,nfracg)	6(1x,e12.5)	
	(ibkey(i,j),j=1,2),i=1,nfracg)	5(i5,i6,4x)	
	(inkey(i,j),j=1,2),i=1,nfracg)	5(i5,i6,4x)	
	(inside(j,i),j=1,2),i=1,nbpt)	5(i5,i6,4x)	
	((inte2(j,i),j=1,2),i=1,nbpt)	5(i5,i6,4x)	
	inte3(1)	9x,i6	
	(inte3(i),i=2,inte3(1))	15i5	
	((inte1(j,i),j=1,6),i=1,nbpt)	6f10.4	
6	iray,idate,title	unformatted	8
	oray,odate,title2	sequential file	
	nchst,nchan,nsets,nf		
	((isetch(is,i),i=1,10),is=1,nsets)		
	(ichsi(i,j),j=1,8),i=1,nchst)		
	((rchsi(i,j),j=1,10),i=1,nchst)		
	((chan(i,j),i=1,nchan),j=1,10) by batches		
	(ifrach(i),i=1,nchan) of 1000		
	((ichan(j,i),j=1,2),i=1,nf)		
7	ibcc	10x,i5	1
	ndir(3)	10x,3i5	
	delphi(3)	10x,3f10.4	
	or		
	seth(6,maxd)	10x,6f10.4	

Table 2-2. CHANGE Input Control

Program Mode	Value of icont	Input groups *
Generation only	1	1,2,3,4(repeat 4 for each channel set)
Generation and mesh	2	1,2,3,4,5,7 (repeat 4 for each channel set and 7 for each flow region)
Mesh only, use previous generation	3	1,6,7 (repeat 7 for each flow region)

*See Table 2.1 for input groups.

Table 2-3. Description of CHANGE Input Variables

Variable	Description
const	Constant value for orientation angle, radius or aperture
delphi(3)	Gradient delta phi if ibcc = 1; up to three values corresponding to flow directions ndir(3)
dseed	Seed for random number generator, type real*8
ev	Expected value of statistical distribution
ibcc	Boundary condition type = 0 use previous boundary conditions for next parallelepiped = 1 constant gradient = 2 set head for each boundary plane = 3 set total flux in "hole" = EOF use previous boundary conditions for all subsequent parallelepipeds
icent	Code for determining channel characteristics, by set = 1 - produce an even grid of channels on each fracture by defining constant orientation and radius and either set aperture to a constant value or statistically generate it. = 2 - statistically generate coordinates of fracture center; either set orientation angles, radius, and aperture to constant values or statistically generate them
ichar	Code for determining individual fracture characteristics; ichar is read for each of orientation, radius and aperture = 0 - transmissivities will be read in instead of aperture = 2 - read constant value = 3 - statistically generate values according to idist = 4 - generate correlating with log of length = 5 - generate correlating to length = 6 - generate using fracture disc aperture as mean
icont	Code for controlling program execution = 1 - generate primary fracture system only = 2 - generate primary fracture system and mesh = 3 - read primary fracture system from local file and generate mesh

idens	Code for specifying number of channel = 0 - channel density specified = 1 - number of channels per fracture specified
idist	Code for distribution to be used for generating values = 1 - normal distribution = 2 - lognormal distribution = 3 - exponential distribution = 4 - gamma distribution (disabled) = 5 - uniform distribution over specified range (other distribution options may be added to fit field data)
iplot	Code for producing input files for plot programs = 0 - no plot files = 1 - generate plot files in RENUM only, after network optimization = 2 - generate plot files in CHANGE, and then in RENUM
iranf	Code for duplicating random number generation of previous run = 0 - random number generation = 1 - read dseed to duplicate previous run
isetch(maxset,10)	For each fracture disc set iset, identification numbers of the channel set(s) to be generated on the discs from set iset. Up to ten channel sets may be used for each fracture set. If isetch(iset,i) = 0, it is assumed that isetch(iset,j) = 0 for all $j \geq i$
iunits	Type of units to be used = 1 cgs = 2 mks
ndir	flow directions in the case ibcc = 1 = 1 inflow through face 1, outflow through face 3 = 2 inflow through face 5, outflow through face 6 = 3 inflow through face 2, outflow through face 4
nchst	Number of channel sets
rlamb	Number of channels per unit area on the disks
sd	Standard deviation of statistical distribution, or half range for uniform distribution (idist = 5)
slope	Parameter for normal distribution relating one characteristic

	with an other ichar = 4 or 5
seth(6,maxd)	If ibcc = 2, set head values for the six flow region boundary planes If ibcc = 3, seth (1,n) contains the total imposed flux for the given parallelepiped Up to three sets of values corresponding to flow directions ndir(3)
title(2)	Two lines of 80 character titles for labeling output
ycept	Parameter for normal distribution correlating one characteristic with an other, ichar=4 or 5

Table 2-4 lists the Input/Output units used by the program, describes the contents of each file, and identifies the subroutines that read or write the data.

Three sample input files, corresponding to the three modes of program operation, are given following Table 2-4. Variable labels are included in input files for the convenience of the user; they are not read by the program.

Further information about the program is given in Appendix A: a description of the program variables and arrays (Tables A-1 and A-2 - the information in Table 2-3 is also repeated here for ease of reference); a subroutine outline (Table A-3) and descriptions of the program subroutines (Table A-4). Appendix B is a listing of the code, and Appendix C contains the input files corresponding to the sample run described in the appendix of the companion report.

Table 2-4. CHANGE Input/Output Files

Unit	Name	Read/ Write	Main Prog./ Subroutine	Description
1	CHANGE.INP	Read	CHANGE RCHAN	Input data Group 1* Input data Groups 3, 4, and 7*
4	RENUM01.DAT RENUM02.DAT . .	Write	WRENUM	Input data for mesh optimization program
5	FRAC3D.DAT	read	RFRACT	specification of the primary fracture system
6	CHANGEPR.DAT	Write	CHANGE	Title, program options, error messages
7	LINES01.DAT LINES02.DAT . .	Write	WLINES	Input data for plotting program
8	CHAN.DAT	Write Read	CHANGE CHANGE	All primary channel system data: file is written when icont = 1 or 2, then read in for each flow region in the run, or in subsequent runs when icont = 3, input data Group 8*
9	CHANGE01.DAT CHANGE02.DAT . .	Read	RINTER	Specification of disc mesh in flow region, one file for each flow region set
10	FIN.DAT	Read	CHAGEN	Created by FMG3D to make sure each hole is connected by a channel to each fracture which intersects it. Will always read file if it exists.

* see Table 2-1 for input groups

3.0 REFERENCES

Billaux, D., 1988. CHANGE: A Numerical Model for Three-Dimensional Modelling of Channelized Flow in Rock. Theory and Design, Lawrence Berkeley Laboratory, Report No. 24910.

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., J. Peterson, S. Bodea 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.

Appendix A

Program Organization and Arrays

Table A-1. Description of CHANGE Program Variables

Name	Description	How Assigned
const	Constant value for orientation angle, radius or aperture	Read by RCHAN
dseed	Seed for random number generator, type real*8	Read by Main
ev	Expected value of statistical distribution	Read by RCHAN
ibc,ibcc	Boundary condition type: = 1 constant gradient, i.e. wedge shaped imposed head = 2 set head for each boundary plane = 3 set flux for each boundary plane	Read by BNDCON
icent	Code for determining channel characteristics, by set = 1 - produce an even grid of channels on each fracture by defining constant orientation and radius and either set aperture to a constant value or statistically generate it. = 2 - statistically generate coordinates of fracture center; either set orientation angles, radius, and aperture to constant values or statistically generate them icent must be set to 2. It has been left to provide flexibility if more options are added in the future.	Set by RCHAN, Reassigned by CHAGEN
ichar	Code for determining individual fracture characteristics; ichar is read for each of orientation, radius and aperture = 2 - read constant value = 3 - statistically generate values according to idist = 4 - generate correlating with log of length = 5 - generate correlating to length = 6 - generate using fracture disc aperture as mean	Set by RCHAN, Reassigned by CHAGEN
icont	Code for controlling program execution = 1 - generate primary fracture system only = 2 - generate primary fracture system and mesh = 3 - read primary fracture system from local file and generate mesh	Read by main
idist	Code for distribution to be used for generating values = 1 - normal distribution	Read by RCHAN

	= 2 - lognormal distribution = 3 - exponential distribution = 4 - gamma distribution (disabled) = 5 - uniform distribution over specified range (other distribution options may be added to fit field data)	Reassigned by CHAGEN
ikeep	Flag for discarding non-conducting pipes = 0 discard dead-ends and isolated clusters = 1 discard isolated clusters and keep dead-ends = 2 keep dead-ends and isolated clusters	Read by RFRACT
inch	Upper limit of the number of channels in the flow region. The fracture disc intersections are then numbered starting at inch+1	Set by LIMIT
inst	Starting location, minus one, of intersections in arrays inte1,inte2 and inside	Read by RINTER
iplot	Code for producing input files for plot programs = 0 - no plot files = 1 - generate plot files in RENUM only, after network optimization = 2 - generate plot files in CHANGE, and then in RENUM	Read by main
iranf	Code for duplicating random number generation of previous run = 0 random number generation = 1 read dseed to duplicate previous run	Read by main
itole	Number of decimal places in tolerance	Read by RFRACT
iunits	Type of units to be used = 1 cgs = 2 mks	Read by main
maxcha	Maximum number of channels, used for dimensioning arrays	Set in parameters statement in main program
maxd	Number of different sets of boundary conditions	Calculated by BNDCON
maxfrc	Maximum number of fractures, used for dimensioning arrays	Set in parameter statement in main program
maxhed	Maximum number of boundary channels, used for dimensioning arrays	Set in parameter statement in main program

maxint	Maximum number of fracture intersections, used for dimensioning arrays	Set in parameter statement in main program
maxnod	Maximum number of pipes intersections, used for dimensioning arrays	Set in parameter statement in main program
maxset	Maximum number of channels sets, used for dimensioning arrays	Set in parameter statement in main program
nbcha	Number of boundary channels	Calculated by LIMIT, modified by DISCHA
nbhol	Value used in making fracture discs into rectangles, through the use of holes	
nbpt	Number of disc intersections, boundary + internal	Read by RINTER
nchan	Number of channels	Set by CHAGEN modified by LIMIT, INTERS, MOVE and DISCHA
nchst	Number of channel sets	Read by Main
ndir	flow directions in the case ibcc = 1 = 1 inflow through face 1, outflow through face 3 = 2 inflow through face 5, outflow through face 6 = 3 inflow through face 2, outflow through face 4	Read by BNDCON
nelem	Number of elements in finite element mesh	Set by DISCHA
nfracb	Number of boundary fractures	Read by RINTER
nfrac	Number of fracture in flow region	Read by RINTER
nfracg	Number of fractures generated by FMG3D, on which channels are generated.	Read by RINTER
nhole	Number of inner boundary parallelepipeds	Read by RINTER
ninb	Number of boundary intersections	Read by RINTER
nnod	Number of internal nodes (pipes intersections)	Set by INTERS modified by DISCHA

nnodes	Total number of nodes in finite element mesh: number of internal nodes plus number of boundary nodes	Set by DISCHA
rlamb	Number of channels per unit area on the disks	Read by RCHAN Reassigned by CHAGEN
sd	Standard deviation of statistical distribution, or half range for uniform distribution (idist = 5)	Read by RCHAN Reassigned by CHAGEN
slope	Parameter for normal distribution relating one characteristic with an other ichar = 4 or 5	Read by RCHAN Reassigned by CHAGEN
nsets	Number of fracture sets	Read by RFRACT
xmesh,ymesh, zmesh,rophi, rothe,rgene	Geometry of generation region and outer flow region. Not used but passed on to next program RENUM.	Read by RINTER. Written by WRENUM.
ycept	Parameter for normal distribution correlating one characteristic with an other, ichar=4 or 5	Read by RCHAN Reassigned by CHAGEN

Table A-2. Description of CHANGE Program Arrays

Name	Description	How Assigned
chan(maxcha,9)	Channel characteristics n=1,nchan chan(1,n) = inclination (in local fracture disc) chan(2,n) = length chan(3,n) = aperture	Assigned during generation
	chan(4,n) = x_c channel center coordinates chan(5,n) = y_c in CHAGEN	Assigned during generation
	or	
	chan(4,n) = x_1 local channel chan(5,n) = y_1 endpoints chan(6,n) = x_2 coordinates (2-D) chan(7,n) = y_2 until WRENUM	Computed by ENDPTS
	or	
	chan(4,n) = x_1 global channel chan(5,n) = y_1 endpoints chan(6,n) = z_1 coordinates (3-D) chan(7,n) = x_2 in WRENUM chan(8,n) = y_2 and chan(9,n) = z_2 WLINES	Computed by WRENUM
delphi(3)	Gradient delta phi if ibcc = 1; up to three values corresponding to flow directions ndir(3)	Read by BNDCON
frac(12,maxfrc)	Fracture characteristics n=1,nfrac frac(1,n) = phi orientation angles frac(2,n) = theta frac(3,n) = radius frac(4,n) = aperture frac(5,n) = x_c frac(6,n) = y_c coordinates of fracture center frac(7,n) = z_c frac(8,n) = area coefficients of fracture plane equation, ax + by + cz + d = 0	Read by RFRAC, then by RINTER

	frac(9,n) = a frac(10,n) = b frac(11,n) = c frac(12,n) = d	
frcoeff(4,60)	Coefficients of flow region boundary plane equations For each plane i, frcoef(1,i) = ai frcoef(2,i) = bi frcoef(3,i) = ci frcoef(4,i) = di such that: ai x + bi y + ci z + di = 0 if (x,y,z) is on plane i. The first six planes define the outer flow region. The inner boundary parallelepipeds are then defined	Read by Subroutine RINTER
hd(6,maxhed)	Values of imposed head or flux at both endpoints of a boundary channel, for up to three different sets of boundary conditions	Calculated by BNDCON
ib(2,maxhed)	Type of boundary node, for both endpoints of boundary channel ibch = 1,nbcha; j=1,2 ib(j,ibch) = 1 if head is imposed at endpoint j ib(j,ibch) = - 1 if flux is imposed at endpoint j ib(j,ibch) = 0 if endpoint j is not a boundary node	Calculated by BNDCON
ibkey(maxfrc,2)	Key to boundary nodes on each fracture, i= 1,nfrac ibkey(i,1) = number of boundary intersections on fracture i ibkey(i,2) = starting position of these intersections in inte1, inte2 and inside	Read by RINTER
iboun(2,maxhed)	Number of the disc boundary intersection(s) that truncate a boundary channel ibch = 1,nbcha iboun(1,ibch) = number of boundary intersection on first endpoint, if it exists iboun(2,ibch) = number of boundary intersection on second endpoint, if it exists	Calculated by LIMIT
ichan(2,maxfrc)	Key to channels on each fracture n=1, nfrac	Set by CHAGEN modified by

	ichan(1,n) = starting position of channel numbers on fracture disc n ichan(2,n) = number of channels on fracture disc n	LIMIT Modified by MOVE
ichsi(maxset,8)	Code for origin of channel characteristics i = 1, nchst, all elements not necessarily filled ichsi(i,1) = icent (only one option now; 2 ... generate) ichsi(i,2) = number of fractures per set ichsi(i,3) = ichar (orientation) ichsi(i,4) = idist (orientation) ichsi(i,5) = ichar (length) ichsi(i,6) = idist (length) ichsi(i,7) = ichar (aperture) ichsi(i,8) = idist (aperture) with ichar = 2 set to constant = 3 generate according to idist = 4 generate correlating with log of length = 5 generate correlating to length = 6 generate using fracture disc aperture as mean idist = 1 normal distribution = 2 lognormal distribution = 3 exponential distribution = 4 gamma distribution (DISABLED) = 5 uniform distribution	Read or set by RCHAN or RCHAGEN
ifrach(maxcha)	Fracture disc on which lies a channel n=1,nchan ifrach(n) = identification number of disc on which the channel lays	Set by CHAGEN Modified by LIMIT
inew(maxfrc)	Back reference array for fracture numbers n = 1,nfracg inew(n) = ifrac, such that iref(ifrac) = n	Set by RINTER
inext(0:maxcha)	Pointer array for pipes i=0,inch + nint inext(i) = number of pipe following pipe i in list of pipes	Set by LIMIT and INTERS modified by DISCHA
inkey(maxfrc,2)	Key to intersections on each fracture n=1,nfrac inkey(n,1) = number of intersections on fracture n inkey(n,2) = starting position of identification numbers of intersections on fracture n in catalog array inte3	Read by RINTER Read by RINTER

inside(2,maxint)	Side codes for endpoints of disc intersections i=1, inst + nint ibside(1,i) = number of boundary plane for endpoint 1 ibside(2,i) = number of boundary plane for endpoint 2	Read by RINTER
inte1(6,maxint)	3-D coordinates of fracture intersection endpoints k=1,ninst+nint, real inte1(1,k) = x_1 inte1(2,k) = y_1 inte1(3,k) = z_1 inte1(4,k) = x_2 inte1(5,k) = y_2 inte1(6,k) = z_2	Read by RINTER
inte2(2,maxint)	Identification numbers of the two fracture discs involved in a fracture-fracture intersection or of the fracture and boundary side involved in a fracture-boundary intersection k=1,inst+nint inte2(1,k) = fracture number inte2(2,k) = fracture number or opposite of side number	Read by RINTER
inte3(maxint*2+1)	Intersection/fracture catalog identifying which intersections are on each fracture j=2,inte3(1) entries in this array are keyed by array inkey inte3(1) = number of entries in the array inte3(j) = intersection identification number	Read in RINTER Modified in LIMIT
inte4(8,maxint)	Local 2-D coordinates of fracture intersections endpoints, in the two fractures they involve k=1,inst+nint,real inte4(1,k) = x_1 inte4(2,k) = y_1 on first fracture inte4(3,k) = x_2 inte4(4,k) = y_2 inte4(5,k) = x_1 inte4(6,k) = y_1 on second fracture inte4(7,k) = x_2 inte4(8,k) = y_2	Calculated by LIMIT
iref(maxfrc)	Reference array for fracture numbers i=1,nfrac iref(i) = old number ifold of fracture number i when it was generated. The characteristics of fracture i, its intersections, are found in location ifold in the appropriate arrays	Read by RINTER

isetch(maxset,10)	Identification numbers of channel sets on each fracture set i = 1, nsets, j = 1, 10 isetch(i,j) = jth identification number of a channel set on fracture disc set number i. Up to ten channel sets can be specified for each fracture set. If less than 10, isetch is filled with trailing zero's.	Read by RCHAN
isetfr(maxfrc)	Number of the set for each fracture disc i=1,nfracg isetfr(i) = number of set from which fracture disc i was generated by FMG3D	Read by RFRACT
jnod(2,maxnod)	Numbers of two pipes making up a pipe intersection. i = 1, nnod jnod(1,i) = first pipe making up intersection number i jnod(2,i) = second pipe	Set by INTERS Modified by DISCHA
knod(maxnod*2)	Array of pipe intersections, keyed by array nodk. i = 1,2*nnod knod(i) = intersection identification number	Set by INTERS Modified by DISCHA
ndir(3)	Flow directions, up to three = 1 inflow through face 1, outflow through face 3 = 2 inflow through face 5, outflow through face 6 = 3 inflow through face 2, outflow through face 4	Read by BNDCON
nodk(2,maxcha)	Key to pipe intersections array nodk on each pipe i = 1, inch + nint nodk(1,i) = starting position of intersection numbers on pipe i nodk(2,i) = number of intersections on pipe i	Set by INTERS Modified by DISCHA
rchsi(maxset,10)	Array used to pass expected values, standard deviations, or constant values for the various parameters defining each channel set i = 1,nset, all elements not necessarily filled rchsi(i,1) = constant orientation rchsi(i,2) = expected value of orientation rchsi(i,3) = standard deviation of orientation rchsi(i,4) = constant length rchsi(i,5) = expected value of length rchsi(i,6) = standard deviation of length rchsi(i,7) = constant aperture rchsi(i,8) = expected value of aperture rchsi(i,9) = standard deviation of aperture	Set by RCHAN

	rchsi(i,10) = areal density of channels	
seth(60,3)	Imposed head or flux on a boundary face, ibcc = 2 or 3 i = n1, n2 n1 = 6*ihole - 5 n2 = 6*ihole j = 1, maxd if ibcc = 2 seth(i,j) = value of imposed head on face (i-n1+1) of parallelepiped number ihole, in set of boundary conditions number j if ibcc = 3, seth (n1,j) = value of imposed flux on whole hole in set of boundary conditions number j	Read by BNDCON
tboun(2,maxhed)	Relative coordinate t_b of channel boundary endpoint along the disc boundary intersection that truncates the channel ibch = 1,nbcha tboun(1,ibch) = t_b of channel/boundary intersection for first channel endpoint of it exists tbound(2,ibch) = same for second endpoint so that t_b = 0 if the channel cuts the boundary intersection at this boundary's first endpoint, and t_b = 1 if the channel cuts it at its second endpoint	Calculated by LIMIT
title(2)	Two lines of 80 character titles for labeling output	Read by main
tnod(2,maxnod)	Relative abscissa (0 at first endpoint and 1 at second endpoint) of a pipe intersection i = 1,nnod tnod(1,i) = relative abscissa of intersection i on channel number jnod(1,i) tnod(2,i) = relative abscissa of intersection i on channel number jnod(2,i)	Set by INTERS Modified by DISCHA

Table A-3. Subroutine Outline

Channel Generation					
	CHANGE	RFRACT			
		RCHAN			
		CHAGEN	GRIDXY		
			RANDXY		
			NORMDI		
			DISTRI	NORMAD	GGUBFS
				LOGNOD	
				EXPOND	
				UNIFOD	
		ENDPTS			
Pipe network					
		RINTER			
		LIMIT	ROFRAC		
			RFRCGLO		
			MOVE		
		INTERS			
		DISCHA			
		BNDCON			
		WRENUM			
		WLINES			

Table A-4. Description of CHANGE Subroutines

BNDCON

Reads boundary conditions parameters and computes imposed head or flux values for boundary nodes. Up to three different sets of boundary conditions may be used at the same time.

given: nhole, maxh, maxhed, maxint, frcoef(4,60),
ibound(2,maxhed), inside(2,maxint), inte1(6,maxint),
tboun(2,maxhed), dmesh
reads: ibcc, ndir(3), delphi(3) or seth(6,3)
returns: maxd, hd(6,maxhed), ib(2,maxhed)

CHAGEN

Generates channel characteristics on all the fractures

given: dseed, nfrac, isetfr(maxfrc), isetch(maxset,10),
frac(12,maxfrc), ichsi(maxset,8) itole, rchsi(maxset,16)
returns: ichan(2,maxfrc), nchan, (chan(maxcha,i), i=1,5),
ifrach(maxcha)
uses subroutines: RANDXY
NORMDI GGUBFS
DISTRI NORMAD
LOGNOD
EXPOND GGUBFS
UNIFOD

DISCHA

Discards channels which cannot conduct flow because they have only one intersection (not discarded if ikeep is 1). This is done by adjusting the pointer array inext.

given: ikeep, nnod, nbcha, nchan, inext (0:maxca),
nodk(2,maxcha), iboun(2,maxched), tboun(2,maxhed),
knod(maxnod*2), jnod(2,maxnod), tnod(2,maxnod)
returns: adjusted nchan, inext, nodk, iboun,
tboun, jnod, tnod, nbcha, knod, nnodes, nbnd, nelem

DISTRI

Calls the appropriate distribution routine to generate a parameter and pass it back to chagen

given: idist, n, dseed, ev, sd
returns: a(n)
uses subroutines: NORMAD
LOGNOD
EXPOND GGUBFS
UNIFOD

ENDPTS

Calculates the endpoints of a channel given its center, orientation and length

given: nchan, maxcha,(chan(maxcha,i), i=2,4,5,8,9)
returns: (chan(maxcha,i), i = 4 to 7)

EQLINE

Calculates the coefficients a, b and c of the line on which the channel lies

given: nchan, maxcha,(chan(maxcha,i), i = 1,4,5)
returns: (chan(maxcha,i), i = 8 to 10)

EXPOND

Generates random variables distributed exponentially with expected value ev

given: n, dseed, ev
returns: a(n)
uses subroutine: GGUBFS

FRSTAT

Calculates a mean and standard deviation

given: n, a(n)
returns: ev, sd

GRIDXY

Generates spaced grid of channel centers along specified orientation

given: n,rad,ang
returns: x(n),y(n)

INTERS

Determines pipe/pipe intersections, beginning with pipes intersecting a boundary (level 1), and then pipes intersecting these ones, and so on. For each pipe considered, only the pipes inside the same fracture(s) are screened for intersection. During this process, fracture intersections are considered as pipes, but their processing is kept distinct from the processing of channels.

given: toler, maxcha, maxnod, nbcha, inext(0:maxcha),
lastbn, inch, chan(maxcha,10), ifrach(maxcha),
inte2(2,maxint), inte4(8,maxint), ichan(2,maxfrc),
inkey(maxfrc,2), inte3(maxint*2+1), inte1(6,maxint)
returns: nchan, nint, nnod, inext(0:maxcha), tnod(2,maxnod),
jnod(2,maxnod), nodk(2,maxcha), knod(maxnod*2)

LIMIT

Truncates a channel if one or both of its endpoints fall outside of the fracture on which it was generated. Then deletes the part(s) of the channel lying outside of the flow region. This may cause the channel to be split in two or more smaller channels. The boundary channels are stored, and internal fracture intersections are rotated from global

to local 2-dimensional fracture coordinates and stored.

given: maxhed, itole, nfrac, inst, iref(maxfrc),
ichan(2,maxfrc), nfracg,inew(maxfrc), frac(3,maxfrc),
inkey(maxfrc,2), nfracb, ibkey(maxfrc,3), inte1(6,maxint),
chan(maxcha,10), inte3(maxint*2+1), inte2(2,maxint),
ifrach(maxcha), nhole
returns: nbcha, tboun(2,maxched), ibound(2,maxched),
inch, inext(0:mascha), inte4(8,maxint), lastbn,
adjusted nchan, chan, ifrach, inte3
uses subroutines: ROFRAC
FRACGLO
MOVE

LOGNOD

Generates random variables distributed lognormally with expected value ev and standard deviation sd

given: n, deseed, ev, sd
return: a(n)

MOVE

Moves the information in the array chan(maxcha,10) and updates the pointer ichan(1,maxcha), for all fractures with number greater than ifold. This subroutine is used to make room for boundary channels newly created due to the splitting of channels at an inner boundary.

given: maxcha, maxfrc, ich, ifold1, nleft, chan(maxcha,10),
ichan(2,maxcha), nfracg, inew(maxfrc), nchan
returns: updated ich, nchan, chan, ichan

NORMAD

Generates random variables distributed normally with expected value ev and standard deviation sd.

given: n, dseed, ev, sd
returns: a(n)
uses: GGUBFS

NORMD1

Generates random variables distributed normally with expected value ev and standard deviation sd, where ev is proportional to another parameter or to the log of another parameter

given: n, dseed, ycept, sd, slope, b(n), ichar
returns: a(n)
uses: GGUBFS

RANDXY

Generates random channel centers within a disc

given: n, dseed, rad, itole
returns: x(n), y(n)
uses: GGUBFS

RCHAN

Reads in channel information, for the following channel parameters distributions: Orientation, length, aperture and density

given: nchst, nsets, maxset
reads: isetch(maxset,10), ichsi(maxset,8), rchsi(maxset,16)

RFRACT

Reads all primary fracture system data from unit 5

given: maxfrc
reads: iray, idate, title2(2), nsets, itole, ikeep,
nfrac, frac(12,maxfrc), isetfr(maxfrc)

RFRCGLO

Transforms coordinates from the local fracture system to the global xyz coordinate system if inv = 1, or from the global system to the local system if inv = 2, using the transformation matrix rotf.

given: isiz1, isiz2, xyz(isiz1, isiz2), rotf(3,3), k1, k2, inv
returns: transformed (xyz(k,isiz2), k=k1, k2)

RINTER

Reads intersection and fracture data from program FMG3D output, for a given flow region

given: nflow
reads: nfracg, nfrac, nfracb, nbpt, inst, ninb, rgene,
nhole, nbhol, xmesh, ymesh, zmesh, rophi, rothe,
frcoef(4,60), iref(maxfrc), frac(12,maxfrc),
ibkey(maxfrc,2), inkey(maxfrc,2), inside(2,maxint),
inte2(2,maxint), inte3(maxint*2+1), intel(6,maxint)
returns: everything it reads, plus inew(maxfrc), iold(maxfrc)

ROFRAC

Computes the matrix of rotation from the global system to the local coordinate system determined by the fracture being investigated, computes the new zplane and the new fracture center (dk, 0, zplane)

given: maxfrc, ifold, frac(12,nfrac)
returns: rotf(3,3), zplane, dk

UNIFOD

Generates random variables uniformly distributed, with minimum center - range and maximum center + range

given: n, dseed, center, range
returns: a(n)
uses: GGUBFS

WLINES

Writes the input file(s) containing the global coordinates of the line elements for the plotting program DIMES. WLINES is called only if iplot is 2

given: nflow, nchan, maxcha, inext(0:maxcha), inch,
ifrach(maxcha), chan(maxcha,10), inte2(2,maxint),
intel(6,maxint), inew(maxfrc)
writes output file on unit 7

WRENUM — Writes the input file for the next program in the chain, RENUM

given: nflow, iray, idate, oray, odate, title(2),
title2(2), nchan nfracg, nfrac, nfracb, nbpt,
inst, itole, rgene, xmesh, ymesh, zmesh, rophi,
rothe, ndir, delphi, visc, spgr, nelem, nnodes,
maxd, ikeep, iplot, inext(maxcha), nchan,
chan(maxcha,10), inch, nbcha, iboun(2,maxhed),
inte2(2, maxint), intel(6,maxint), inside(2,maxint),
ifrach(maxcha), inew(maxfrc), nodk(2,maxcha),
frac(12,maxfrc), tnod(2,maxnod), jnod(2,maxnod)
writes: all channel network information needed by program RENUM.

Appendix B

Program Listing

Listing Files and Include File

Each of the 23 subroutines which make up the program is stored in a separate file. The name of the file containing a given subroutine is obtained by adding the suffix ".f" to the name of the subroutine. The file containing the main program is called "change.f". The 24 files constituting the program listing are: bndcon.f, chagen.f, change.f, discha.f, distri.f, endpts.f, expond.f, ggubfs.f, inters.f, limit.f, lognod.f, move.f, normad.f, normd1.f, oiest.f, randxy.f, rchan.f, rfract,f, rfrcglo.f, rinter.f, rofrac.f, unifod.f, wlines.f, wrenum.f.

All the common blocks, array size definitions, and most of the dimensioning statements have been grouped in a file named "change.cmn". An "include" statement is used to insert this file in the subroutine that needs it. The content of change.cmn is printed below

```
      parameter(maxint=500000,maxfrc=10000,maxset=8,maxhed=100000,
     '          maxcha=200000,maxnod=2000000)
      byte ib(2,maxhed)
      common/bounds/tboun(2,maxhed),iboun(2,maxhed),ib,
     '              nbcha,ilast,nhole,nbhol
      common/chanls/chan(maxcha,9),ifrach(maxcha)
      common/flor/frcoef(4,120)
      common/fract/frac(12,maxfrc),rgene,xmesh,ymesh,zmesh,
     '             rophi,rothe,iref(maxfrc),inew(maxfrc),
     '             isetfr(maxfrc),ichan(2,maxfrc),
     '             nfrac,nfracg,nfracb,nsets
      common /head/ hd(6,maxhed),delphi(3),seth(120,3),
     '              ndir(3),maxd
      real*4 inte1(18,maxint+1),inte4(18,maxint)
      integer*4 inte2(18,maxint),inside(18,maxint)
      common/inter/inte1,
     1             inte3(maxint*2+1),ibkey(maxfrc,3),
     2             inkey(maxfrc,2),inst,nbpt,ninb
      equivalence (inte1(18,1),inte2(12,1),inside(10,1),
     1             inte4(8,1))
      common/nextc/inext(0:maxcha),inch
      integer*4 jnod(4,maxnod)
      real*4 tnod(4,maxnod)
      common/nodes/tnod,nodk(2,maxcha),knod(maxnod*2)
      equivalence (jnod(2,1),tnod(4,1))
      common/param/iranf,itole,nchan,ikeep,
     '             nnodes,nnod,nelem,iprnt,itrans
      common/stats/rchsi(maxset,10),ichsi(maxset,10),
     '             isetch(maxset,10),nchst
```

```
      SUBROUTINE BNDCON
c
c *** If boundary condition option ibcc =1 :
c *** subroutine bndcon reads boundary conditions in the form of flow
c *** direction and delta phi, for one, two, or three directions,
c *** and calculates head values for all boundary nodes.  Flow
c *** directions are determined by the value of ndir(i), i=1,3:
c ***   ndir(i) = 1   flow in x direction from side 1 to 3,
c ***   ndir(i) = 2   flow in y direction from side 5 to 6,
c ***   ndir(i) = 3   flow in z direction from side 2 to 4.
c *** The value of head on the inflow side is delta phi and on the
c *** outflow side head is zero.  The direction of flow can be reversed
c *** by using a negative value of delta phi.
c
c *** If ibbc = 2: head values are read in for each flow region
c *** boundary plane and head values set for nodes intersecting
c *** those planes.  Up to three sets of head values can be set.
c
c *** If ibbc = 0: use boundary conditions set from previous flow
c *** region
c
#include "change.cmn"
c
      dimension iside(3,2)
      character*55 btype(3)/
     1' Constant average gradient (wedge shaped imposed heads)',
     2' Set head values at each boundary plane                 ',
     3' Set flux values at each boundary plane                 '/
      data iside/1,5,2,3,6,4/
c
c     loop over boundary domains
c
      do ihole=1,nhole+1
c
c       Read boundary condition option and flow directions,
c       set the number of flow directions.
c
        n1=6*ihole-5
        n2=6*ihole
        ibcc=0
        read (1,110,end=7) ibcc
        if(ibcc.eq.0)goto 7
        if(ibcc.gt.3.or.(ibcc.eq.3.and.nhole.eq.0))goto 95
        ibc=ibcc
        read (1,120,end=93) (ndir(i),i=1,3)
        maxd=0
        if (ndir(1) .ne. 0) maxd=1
        if (ndir(2) .ne. 0) maxd=2
        if (ndir(3) .ne. 0) maxd=3
        if (maxd .eq. 0) go to 97
    7   write (6,200) btype(ibc),maxd
c
c
        if(ibc.eq.1)then
c
c       Read delta phis.
c
          if (ibcc.ne.0) read (1,130,end=93) (delphi(i),i=1,maxd)
          write (6,210)
         else
c
c       Read head or flux values for each boundary plane
c       in maxd directions.
c
```

```
          if (ibcc.gt.0) then
            read (1,160,end=93)((seth(i,j),i=n1,n2),j=1,maxd)
           else
            do j=1,maxd
              do i=n1,n2
                seth(i,j)=seth(i-6,j)
               enddo
             enddo
           endif
          if(ibc.eq.2)then
            write (6,230)
           else
            write (6,240)
           endif
          write (6,250) (j,(seth(i,j),i=n1,n2),j=1,maxd)
         endif
c
        do k=1,maxd
          jk=2*(k-1)
          nin =iside(ndir(k),1)+6*(ihole-1)
          nout=iside(ndir(k),2)+6*(ihole-1)
          dmesh=frcoef(4,nout)-frcoef(4,nin)
          write (6,220) k,nin,nout,delphi(k)
c
          do ibch=1,nbcha
            do 30 iend=1,2
              ibin=iboun(iend,ibch)
              if(ibin.gt.0)then
                ns=inside(iend,ibin)
                if((ns-n1)*(ns-n2).gt.0)goto 30
                if(ibc.eq.1)then
                  ib(iend,ibch)=1
                  hd(jk+iend,ibch)=0.
                  if (ns .eq. nout) go to 30
                  if (ns .ne. nin) go to 20
                  hd(jk+iend,ibch)=delphi(k)
                  go to 30
c
c                 Compute head for a point on a wedge shaped boundary.
c
   20             d=0.0
                  do i=1,3
                    xyz=intel(i,ibin)+tboun(iend,ibch)*
                        (intel(i+3,ibin)-intel(i,ibin))
                    d=d + frcoef(i,nout)*xyz
                   enddo
                  d=abs(d + frcoef(4,nout))
                  hd(jk+iend,ibch)=(d*delphi(k))/dmesh
                 elseif(ibc.eq.2)then
                  ib(iend,ibch)=1
                  hd(jk+iend,ibch)=seth(ns,k)
                 elseif(ibc.eq.3)then
                  ib(iend,ibch)=-1
                  hd(jk+iend,ibch)=seth(n1,k)
                 endif
                endif
   30         continue
             enddo
           enddo
         enddo
c
       return
c
c *** Write error messages and stop the program.
```

```
c
   93 write (6,140)
      write (6,135)
      stop
c
   95 write (6,140)
      write (6,145)
      stop
c
   97 write (6,140)
      write (6,150)
      stop
c
  110 format (10x,i5)
  120 format (10x,3i5)
  130 format (10x,3f10.4)
  135 format (36h incomplete boundary condition input)
  140 format (1h0,36h ***ERROR STOP *** subroutine BNDCON)
  145 format (12h ibcc .gt. 2)
  150 format (30h no flow directions were input)
  160 format (10x,6f10.4)
  200 format (1h0,28h *** BOUNDARY CONDITIONS ***,//,a55,1h,,/,
     1  20h flow is computed in,i2,12h directions.,/)
  210 format (1h ,36h  flow      inflow   outflow   delta,/,
     1            36h direction    side     side      phi,/)
  220 format (i6,i11,i9,f11.4)
  230 format (1h ,35h  flow     heads at boundary planes,/,
     1 11h  direction,4x,1h1,9x,1h2,9x,1h3,9x,1h4,9x,1h5,9x,1h6,/)
  240 format (1h ,35h  flow      flux at boundary planes,/,
     1 11h  direction,4x,1h1,9x,1h2,9x,1h3,9x,1h4,9x,1h5,9x,1h6,/)
  250 format (i7,1x,6f10.4)
c
      end
```

```
c
      subroutine chagen (dseed)
c
c     ****************************************************************
c
c     this subroutine  generates the following channel characteristics:
c
c     orientation, length, aperture, and the coordinates of the
c     channel  center, in the local coordinates of each fracture.
c
c     the following variables or arrays from the main program are
c     used ---
c
c       nchst = number of channel  sets,
c
c       area of the fracture, frac(8,i)
c
c       chan(i, 1) = orientation,
c       chan(i, 2) = length,
c       chan(i, 3) = aperture,
c       chan(i, 4) = x1, x coordinate of channel  center or end,
c       chan(i, 5) = y1, y coordinate of channel  center or end.
c       chan(i, 6) = x2, x coordinate of channel  end,
c       chan(i, 7) = y2, y coordinate of channel  end.
c       chan(i, 8) = a, coeff. = -sin(orientation)
c       chan(i, 9) = b, coeff. =  cos(orientation)
c       chan(i,10) = c, coeff. = -(a*x1 + b*y1)
c
c     the 'randomness' of the generation is controlled by iranf and
c     dseed ---
c
c       iranf = 0 - *random* generation,
c       iranf = 1 - use dseed from previous generation,
c
c       dseed - seed for random number generator.
c
c     the variables which control the generation are given
c     below ---
c
c
c       ichar - read value for each of orientation, length and
c               aperture,
c             = 2 - set characteristic equal to a constant value,
c             = 3 - generate characteristics values according to a
c                   distribution,
c             = 4 - generate apertures correlated to log of length
c             = 5 - generate apertures correlated to length
c             = 6 - generate apertures with different mean for each
c                   fracture
c
c       idist = 1 - normal distribution,
c               2 - lognormal distribution,
c               3 - exponential distribution.
c               4 - gamma distribution
c               5 - uniform distribution.
c
c     in addition, the following variables are read in ---
c
c       const - see ichar = 2,
c
c       ev - expected value of statistical distribution,
c
c       sd - standard deviation of statistical distribution.
c
```

```
c     the array ichsi is used to store the values of nchan, icent, and
c     ichar and idist for each set.  the array rchsi is used to store
c     the values of const, ev and sd for each set.
c
c     ********************************************************************
c
c
#include "change.cmn"
      double precision dseed
      dimension xyz(6,1),rotf(3,3)
c
      parameter (pi=3.141596)
      pi180=pi/180.
c
c
c *** if iranf = 0, pick a random dseed
c
      if (iranf .eq. 0) dseed = secnds(0.0) * 100.0
      write(6,80)dseed
c
      n1=10*nfrac
      do ifrac=1,nfrac
        iset=isetfr(ifrac)
        phi=frac(1,ifrac)
        theta=frac(2,ifrac)
        call ROFRAC(ifrac,rotf,zplane,dk)
        ichn=1
        ichan(1,ifrac)=n1
        ichfr=0
c
        do while (isetch(iset,ichn).ne.0)
          i=isetch(iset,ichn)
c
          rlamb=rchsi(i,10)
          idens = ichsi(i,9)                              !jep 29sep88
          nchan = int(rlamb)                              !jep 29sep88
          if(idens.eq.0)nchan=int(frac(3,ifrac)*frac(3,ifrac)*pi*rlamb)
          n2=n1+nchan-1
          if(n2.gt.maxcha)then
            stop 'too many channels generated'
           endif
          ichsi(i,2)=ichsi(i,2)+nchan
          ichfr=ichfr+nchan
c
          do j=n1,n2
            ifrach(j)=ifrac
           enddo
c
c  Following lines inserted by jep 19jul88 to generate an even grid
c
          icent = ichsi(i,1)
          if(ichsi(i,3).lt.3) ang = rchsi(i,1)
          if(ichsi(i,3).ge.3) ang = rchsi(i,2)
          if(icent.eq.1) call gridxy(chan(n1,4),chan(n1,5),n2-n1+1,
     1                frac(3,ifrac),ang)
          if(icent.eq.2) call randxy(chan(n1,4),chan(n1,5),n2-n1+1,
     1                dseed,frac(3,ifrac),itole)
c
c ********************************************************************
          l=1
          m=-2
          do k=1,3
            l=l+2
            m=m+3
```

```
c
            ichar=ichsi(i,1)
c
c           adjust mean orientation on disc to get the
c           right orientation in 3-D:
c
c           rotate channels on the fracture so that the 3-D strikes of
c           the channels in different fractures of the same set have
c           the same mean given by the user.
c
            if(k.eq.1)then
              if (ichar.lt.3) then
                ev=rchsi(i,m)
               else
                ev=rchsi(i,m+1)
               endif
              vert=90.-phi
              if(ichar.ne.1.and.abs(vert).ge.1.)then    !jep 19jul88
                themb=theta-ev
                themb1=themb*pi180
                phi1=phi*pi180
                ev1=ev*pi180
                xyz(1,1)=cos(ev1)
                xyz(2,1)=-sin(ev1)
                xyz(3,1)=-tan(phi1)*cos(themb1)
                call rfrcglo(6,1,xyz,1,1,rotf,2)
                if(abs(xyz(1,1)).ge.1.e-4)then
                  ev=atan(xyz(2,1)/xyz(1,1))
                  ev=ev/pi180
                 endif
               endif
             endif
c
c           set channel  characteristic to a constant
c
            if (ichar.lt.3) then
              if(k.eq.1)then
                if(ichar.eq.1) ev = 90. - ev        ! jep 19jul88
                const=ev
               else
                const=rchsi(i,m)
               endif
              do j=n1,n2
                chan(j,k)=const
               enddo
c
             elseif(ichar.eq.4.or.ichar.eq.5)then
              ycept=rchsi(i,m)
              slope=rchsi(i,m+1)
              sd=rchsi(i,m+2)
              call normd1(chan(n1,k),n2-n1+1,dseed,ycept,sd,slope,
     1                    chan(n1,2),ichar)
             else
              idist=ichsi(i,1+1)
              if(ichar.eq.3)then
                if(k.ne.1)ev=rchsi(i,m+1)
                sd=rchsi(i,m+2)
               elseif(ichar.eq.6)then
                ev=frac(4,ifrac)
                sd=ev*rchsi(i,m+2)
               endif
              call distri(idist,chan(n1,k),n2-n1+1,dseed,ev,sd)
             endif
           enddo
```

```
c
          n1=n2+1
          ichn=ichn+1
         enddo
        ichan(2,ifrac)=ichfr
       enddo
      nchan=n2
      return
c
   80 format('0the initial seed used in the random number generator',
     1  ' is ',d15.8)
      end
```

```
      program change
c     ************************** s e t u p ****************************
c
#include "change.cmn"
      common /qc/ visc,spgr,qc
c
      character*80 title(2),title2(2)
      character*9 idate,odate
      character*19 iray,oray
      character*3 cgsmks(2)
      double precision dseed
      data cgsmks/'cgs','mks'/
c
c *** open files
c
      open (unit=1,file='change.inp',status='old')
      open (unit=6,file='changepr.dat',status='unknown')
c
c
c *** request job information.
c
      iray = 'change Version 1.00'
      call date (idate)
c
c     ************************** i n p u t ****************************
c
c *** read control variables
c
c
      read (1,260) icont,iplot,iunits,iprnt
      read (1,270) iranf,dseed
      read(1,330)title
      write(6,330)title
      write(6,290) icont,iplot
c
c
      if (iunits.eq.0) then
        visc=.011
        spgr=980.66
       else
        visc=.0011
        spgr=9806.6
       endif
      qc=spgr/(12.*visc)
c
      nflow=0
      if (icont.eq.3) go to 50
c
c *** read input variables
c
      read (1,280) nchst
c
c     call subroutine RFRACT to read fracture characteristics
c     from FMG3D
c
      call rfract(oray,odate,title2)
c
c     call subroutine RCHAN to read channel sets characteristics.
c
      call rchan
c
c     ********** c h a n n e l    g e n e r a t i o n **************

c
```

```
c *** call subroutine CHAGEN to generate channel
c *** characteristics and store them in arrays chan, ifrach,
c *** ichan.
c
      call chagen (dseed)
c
c *** program stop if nchan greater than maxcha
c
      if (nchan.gt.maxcha) then
        write(6,140) nchan,maxcha
        stop
       endif
c
c *** print the contents of chan.
c
      write(6,210) iray,idate
      write(6,450)
      write(6,470) nchan
      do 30 m=1,nchst
        write(6,220)
        write(6,230) m,ichsi(m,2),rchsi(m,10)
        write(6,240)
   30   continue
c
c *** fill array chan ---
c *** 1.  call subroutine eqline to calculate coefficients of the
c ***     equation of the line on which the channel  lies,
c ***     LINE EQUATIONS ARE NOT USED ANY MORE.  SKIP EQLINE.
c *** 2.  call subroutine endpts to calculate channel  endpoints,
c
c     call eqline
      call endpts
c
c *** calculate and print channel  statistics for each set
c
      write(6,540)
c      call pfs
c
c     write generated channels on unit 8
c
      open (unit=8,file='chan.dat',form='unformatted')
      write (8) iray,idate,title
      write (8) oray,odate,title2
      write (8) nchst,nchan,nsets,nfrac
      write (8) ((isetch(is,i),i=1,10),is=1,nsets)
      write (8) ((ichsi(i,j),j=1,8),i=1,nchst),
     1          ((rchsi(i,j),j=1,10),i=1,nchst)
      do i1=1,nchan,1000
        i2=min0(nchan,i1+999)
        write (8) ((chan(i,j),i=i1,i2),j=1,10)
        write (8) (ifrach(i),i=i1,i2)
       enddo
      write (8) ((ichan(j,i),j=1,2),i=1,nfrac)
      close (unit=8)
c
      if (icont.eq.1) then
        write(6,150)
        stop
       endif
c
c *************** CHANNEL NETWORK **********************
c
  50  nflow=nflow+1
c
```

```
c     call subroutine RINTER to read intersection information
c     from FMG3D
c
      call rinter (nflow)
c
c *** read data from unit 8
c
      open (unit=8,file='chan.dat',status='old',form='unformatted')
c
      read (8) iray,idate,title
      read (8) oray,odate,title2
      read (8) nchst,nchan,nsets,nf
      read (8) ((isetch(is,i),i=1,10),is=1,nsets)
      read (8) ((ichsi(i,j),j=1,8),i=1,nchst),
     1          ((rchsi(i,j),j=1,10),i=1,nchst)
      do i1=1,nchan,1000
        i2=min0(nchan,i1+999)
        read (8) ((chan(i,j),i=i1,i2),j=1,10)
        read (8) (ifrach(i),i=i1,i2)
       enddo
      read (8) ((ichan(j,i),j=1,2),i=1,nf)
      close (unit=8)
c
c
c ***     call subroutine limit to truncate any channel falling
c ***     outside its fracture or the flow region.
c
      call limit
c
c *** print the contents of chan and kut.
c
      write(6,210) iray,idate
      write(6,480)
      write(6,470) nchan
c
c *** calculate and print channel  statistics for each set
c
      write(6,550)
c      call pfs
c
c     call subroutine INTERS to calculate channels intersections
c
      call inters
c
c     call subroutine DISCHA to eliminate channels with only one
c     intersection if ikeep is less than 1
c
      call discha
c
c     call subroutine BNDCON to read specifications for boundary
c     conditions and calculate them at each boundary node.
c
      call bndcon
c
c     call subroutine WRENUM to write input file for finite element
c     program
c
      call wrenum(nflow,iplot,iray,idate,title,oray,odate,title2)
c
c     call subroutine WLINES to write input file for plotting program,
c     if iplot=2
c
      if (iplot.eq.2)call wlines(nflow)
      go to 50
```

```
c
  130 write(6,160)
      stop
c
c     *************** f o r m a t   s t a t e m e n t s ****************
c
c
c
  140 format('1',39('-')/
     1  ' program stop,nchan is greater than maxcha,nchan = ',i6/
     2  ' maxcha= ',i6/1x,55('-'))
  150 format('0normal program stop,icont=1')
  160 format('0normal program stop,end of input')
  170 format('0no channels in flow region for rotan=',f6.2)
  180 format('0there are no conducting channels for rotan=',f6.2)
  190 format('0normal end of generation,imesh =0')
  200 format(1x)
  210 format('0',a19,' - ',a9)
  220 format('0',10(/),6x,'set      number of channels',6x,
     1  'density of channels'/)
  230 format('0',5x,i5,15x,i5,15x,e10.4)
  240 format(10(/))
  250 format(2(10x,f10.4),10x,i5)
  260 format(10x,i5,4(15x,i5))
  270 format(10x,i5,15x,d15.8)
  280 format(3(10x,i5))
  290 format('0icont=  ',i5,10x,'iplot =  ',i5)
  300 format(a7,' --- ',a9)
  310 format(4i10)
  320 format(10x,i5,2(10x,f10.4))
  330 format(a)
  340 format(10x,i5)
  350 format(8i10)
  370 format(5e12.4)
  380 format(2f10.4,e10.4,4f10.2,3x,z2)
  390 format(6f10.2)
  400 format(4f10.4)
  410 format('*** units are ',a,' ***')
  420 format('--nchst---',i5)
  430 format('--nchan---',i5,'--xgene---',f10.4,'--ygene---',f10.4)
  440 format(3(10x,f10.4),7x,i1)
  450 format('0f r a c t u r e   g e n e r a t i o n')
  460 format('0the size of the generation region is ',f8.1,' by ',f8.1/
     1  ' the number of subregions is ',i3)
  470 format('0the number of channels generated or read in is ',i5)
  480 format('0t r u n c a t e d   f r a c t u r e s   o f',
     1  '   g e n e r a t i o n   s t a g e')
  490 format('0f r a c t u r e s   i n   f l o w   r e g i o n')
  500 format(' ( a l l   f r a c t u r e s   i n c l u d e d )')
  510 format('0the size of the flow region is ',f8.1,' by ',f8.1/
     1  ' the angle of rotation is ',f6.2)
  520 format('0( n o n - i n t e r s e c t i n g   f r a c t u r e s',
     1  '   h a v e   b e e n   d r o p p e d )')
  530 format('0the number of channels in the flow region is ',i5)
  540 format('0f r a c t u r e   s t a t i s t i c s')
  550 format('0f r a c t u r e   s t a t i s t i c s'/
     1  ' o f   t r u n c a t e d   f r a c t u r e s')
  560 format('0f r a c t u r e   s t a t i s t i c s'/
     1  ' o f   f r a c t u r e s   i n   f l o w   r e g i o n')
  570 format('0isolated channels have been eliminated')
  580 format((4(1x,2g9.3,1x)))
  590 format(' view'/2x,2(14x,'0.',14x,'1.'))
  600 format(10x,4(6x,g10.4))
  610 format(' tick'//' limi')
```

```
620 format(' dash'/' number of points=',i4/)
    end
```

```
change.cmn
2c2
<      '           maxcha=50000,maxnod=20000)
---
>      '           maxcha=100000,maxnod=25000)
5c5
<      '               nbcha,ilast,nhole,nbhol
---
>      '               nbcha,ilast,nhole

*********************************
chagen.f
111d110
< c  Following lines inserted by jep 19jul88 to generate an even grid
113,121c112,113
<            icent = ichsi(i,1)
<            if(ichsi(i,3).lt.3) ang = rchsi(i,1)
<            if(ichsi(i,3).ge.3) ang = rchsi(i,2)
<            if(icent.eq.1) call gridxy(chan(n1,4),chan(n1,5),n2-n1+1,
<      1                  frac(3,ifrac),ang)
<            if(icent.eq.2) call randxy(chan(n1,4),chan(n1,5),n2-n1+1,
<      1                  dseed,frac(3,ifrac),itole)
< c
< c ***************************************************************************
---
>            call randxy(chan(n1,4),chan(n1,5),n2-n1+1,dseed,
>      1                  frac(3,ifrac),itole)
144c136
<               if(ichar.ne.1.and.abs(vert).ge.1.)then   !jep 19jul88
---
>               if(abs(vert).ge.1.)then
164d155
<                 if(ichar.eq.1) ev = 90. - ev       ! jep 19jul88

*****************************
inters.f
19c19
<       dimension inod(500)              !jep 1jul88 to prevent write-overs
---
>       dimension inod(100)

*****************************
limit.f
36c36
<       nhol1=nhole+nbhol+1     !jep 1jul88 - for square fracture simulation
---
>       nhol1=nhole+1

*****************************
rchan.f
76,77c76
< c       only options available now is generate at random
< c       or evenly spaced centers (icent = 1)        jep 19jul88
---
> c       only option available now is generate at random
79c78
<          ichsi(i,1)=icent
---
>          ichsi(i,1)=2

*********************************
rinter.f
17c17
<      '              rgene,nhole,nbhol,xmesh,ymesh,zmesh,rophi,rothe
```

```
---
>                     rgene,nhole,xmesh,ymesh,zmesh,rophi,rothe
51c51
<   900 format (2(10x,i5,15x,i5,15x,i5/),10x,f10.0,10x,i5,10x,i5/
---
>   900 format (2(10x,i5,15x,i5,15x,i5/),10x,f10.0,10x,i5/

*****************************
wrenum.f
11c11
<       dimension inod(500),tkut(500)     !jep 1jul88 to prevent write-over
---
>       dimension inod(100),tkut(100)

*******************************
```

```
c
c
      subroutine discha
c
c     this subroutine discards channels which cannot conduct flow (only
c     one intersection), only if ikeep is 0.
c
#include "change.cmn"
c
      if(ikeep.ne.0)then
        nnodes=nnod+nchan*2
        nelem=nnod*2+nchan
        return
       endif
c
  10  ich=0
      ibch=0
      nelim=0
      nelem=0
      nbnd=0
      nnod=0
      do 30 i=1,nchan
        ilast=ich
        ich=inext(ich)
        nif=nodk(2,ich)
        if(i.le.nbcha)then
          if(iboun(1,i).eq.0.or.iboun(2,i).eq.0)then
            if(nif.eq.0)then
              inext(ilast)=inext(ich)
              ich=ilast
              nelim=nelim+1
              goto 30
             else
              nbnd=nbnd+1
              nelem=nelem+nif
             endif
           else
            nbnd=nbnd+2
            nelem=nelem+nif+1
           endif
          nnod=nnod+nif
          ibch=ibch+1
          do j=1,2
            tboun(j,ibch)=tboun(j,i)
            iboun(j,ibch)=iboun(j,i)
           enddo
         else
          if(nif.eq.1)then
            ient=nodk(1,ich)
            iint=knod(ient)
            jch=jnod(1,iint)
            if(jch.eq.ich)jch=jnod(2,iint)
            njf=nodk(2,jch)
            nod1=nodk(1,jch)
            nod2=nod1+njf-1
            do jent=nod1,nod2
              jint=knod(jent)
              j1=jnod(1,jint)
              j2=jnod(2,jint)
              if(j1.eq.ich.or.j2.eq.ich)then
              if(jent.eq.nod2)goto 20
                do jin=jent,nod2-1
                  knod(jin)=knod(jin+1)
                enddo
```

```
                goto 20
               endif
             enddo
   20       nodk(2,jch)=njf-1
           endif
          if(nif.le.1)then
            nelim=nelim+1
            inext(ilast)=inext(ich)
            ich=ilast
           else
            nnod=nnod+nif
            nch=nch+1
            nelem=nelem+nif-1
           endif
         endif
   30  continue
       nchan=nchan-nelim
       nbcha=ibch
       if(nelim.gt.0)goto 10
c
c      reorder nodes
c
       ich=0
       nnod=0
       do i=1,nchan
         ich=inext(ich)
         nif=nodk(2,ich)
         if(nif.ne.0)then
           ient1=nodk(1,ich)
           ient2=ient1+nif-1
           do 50 ient=ient1,ient2
             iint=knod(ient)
             if(jnod(1,iint).eq.ich)then
               nnod=nnod+1
               knod(ient)=nnod
               do j=1,2
                 tnod(j,nnod)=tnod(j,iint)
                 jnod(j,nnod)=jnod(j,iint)
                enddo
               jch=jnod(2,nnod)
               njf=nodk(2,jch)
               jent1=nodk(1,jch)
               jent2=jent1+njf-1
               do jent=jent1,jent2
                 jint=knod(jent)
                 if(jint.eq.iint)then
                   knod(jent)=nnod
                   goto 50
                  endif
                enddo
              endif
   50        continue
          endif
        enddo
c
       nnodes=nnod+nbnd
c
       return
       end
```

```
c
      subroutine distri (idist,a,n,dseed,ev,sd)
c
c *** this subroutine calls the appropriate distribution routine
c     based upon the idist argument
c     idist = 1 - normal
c             2 - lognormal
c             3 - exponential
c             4 - gamma DISABLED
c             5 - uniform
c
      dimension a(n)
      if (idist*(idist-6).ge.0) stop 'unknown distribution'
      go to (10,20,30,40,50),idist
   10 call normad (a,n,dseed,ev,sd)
      return
   20 call lognod (a,n,dseed,ev,sd)
      return
   30 call expond (a,n,dseed,ev)
      return
   40 stop ' gamma distributions disabled'
c     call gammad (a,n,dseed,ev,sd)
c     return
   50 call unifod (a,n,dseed,ev,sd)
      ev=(ev+sd)/2.
      return
      end
```

```
subroutine dum
x = x + 1
return
end
```

```
      subroutine endpts
c
c     ***********************************************************************
c
c     this subroutine calculates the endpoints of a channel  given the
c     center, orientation and length of the channel .
c
c     nchan is the number of channels.
c
c     the components of chan used in this subroutine are ---
c       chan(i,1) = orientation,
c       chan(i,2) = length,
c       chan(i,4) = xc, x coordinate of channel  center,
c       chan(i,5) = yc, y coordinate of channel  center.
c
c     the components of chan that are calculated in this subroutine
c     are ---
c       chan(i,4) = x1, x coordinate of endpoint 1,
c       chan(i,5) = y1, y coordinate of endpoint 2,
c       chan(i,6) = x2, x coordinate of endpoint 2,
c       chan(i,7) = y2, y coordinate of endpoint 2.
c
c     the endpoint (x1,y1) lies on the ray which forms an angle =
c     chan(i,1) with the positive x-axis.  (x2,y2) lies on the ray
c     forming an angle = chan(i,1) + 180 with the positive x-axis.
c
c     ***********************************************************************
c
c
#include "change.cmn"
c
c     define pi/180.
c
      pi180 = atan(1.)/45.
c
      do 110 i=1,nchan
c
c     set local variables.
c
        orie=chan(i,1)
        xc=chan(i,4)
        yc=chan(i,5)
c
c     convert orie from degrees to radians.
c
        orie=orie*pi180
        a=-sin(orie)
        b=cos(orie)
c
c     calculate coordinates of endpoints
c
        hlen=.5*chan(i,2)
        dx=hlen*b
        dy=hlen*a
        chan(i,4)=xc-dx
        chan(i,5)=yc+dy
        chan(i,6)=xc+dx
        chan(i,7)=yc-dy
c
  110   continue
c
      return
      end
```

```
      subroutine expond (a,n,dseed,ev)
c
c *** this subroutine generates random variables distributed
c *** exponentially with expected value ev.
c
c *** if x is distributed uniformly in (0,1), then y = -ln(1-x)*ev
c *** is distributed exponentially with parameter lambda = 1/ev.
c *** the expected value of y is ev.
c
      dimension a(n)
c
      do 110 i=1,n
        a(i)=-alog(1.-ggubfs(dseed))*ev
  110   continue
c
      return
      end
```

```
c   imsl routine name   - ggubfs
c
c-----------------------------------------------------------------------
c
c   computer            - cray/single
c
c   latest revision     - june 1, 1980
c
c   purpose             - basic uniform (0,1) random number generator -
c                           function form of ggubs
c
c   usage               - function ggubfs (dseed)
c
c   arguments    ggubfs - resultant deviate.
c                dseed  - input/output double precision variable
c                           assigned an integer value in the
c                           exclusive range (1.d0, 2147483647.d0).
c                           dseed is replaced by a new value to be
c                           used in a subsequent call.
c
c   precision/hardware  - single/all
c
c   reqd. imsl routines - none required
c
c   notation            - information on special notation and
c                           conventions is available in the manual
c                           introduction or through imsl routine uhelp
c
c   copyright           - 1978 by imsl, inc. all rights reserved.
c
c   warranty            - imsl warrants only that imsl testing has been
c                           applied to this code. no other warranty,
c                           expressed or implied, is applicable.
c
c-----------------------------------------------------------------------
c
      real function ggubfs (dseed)
c                                 specifications for arguments
      double precision   dseed
c                                 specifications for local variables
      double precision   s2p31,s2p31m,seed
c                                 s2p31m = (2**31) - 1
c                                 s2p31 = (2**31)
      data               s2p31m/2147483647.d0/,s2p31/2147483648.d0/
c                                 first executable statement
      seed = dseed
      seed = dmod(16807.d0*seed,s2p31m)
      ggubfs = seed / s2p31
      dseed = seed
      return
      end
c
```

```
      subroutine gridxy (x,y,n,rad,ang)
c
c *** this subroutine generates evenly spaced channel  centers - jep 19jul88
c
      dimension x(n),y(n)
c
      diam=2.*rad
      srad=rad*rad
c
      xinc = diam/float(n+1)
      xn = -rad
      do i=1,n
        xn = xn + xinc
        x(i)=xn*cosd(ang)
        y(i)=xn*sind(ang)
        dist=x(i)*x(i)+y(i)*y(i)
        if(dist.ge.srad) stop 'gridxy'
       enddo
c
 10   return
      end
```

```
c
c
      SUBROUTINE INTERS
c
c *** This subroutine determines channel/channel
c *** intersections ,beginning with channels intersecting
c *** a side(level 1), and then channels intersecting these ones,
c *** and so on.  In level 1, intersections are truncated at the
c *** flow region boundaries, if necessary.  With this method, only
c *** channels connected to at least one side are considered.
c *** the 'nodes' common is used here only as a temporary memory location.
c *** For each channel considered, only the channel inside
c *** the same(s) fracture(s) are considered for intersections.

c
#include "change.cmn"
      dimension ifirsi(maxcha),ilasi(maxcha),nexti(maxnod)
      dimension xy(8),ifol(2)
      dimension inod(10000)            !jep 1jul88 to prevent write-overs
c
      toler=1./(10**itole)
      stoler=toler*toler
c
c     initialize level one
c
      do i=1,maxcha
        ifirsi(i)=0
        ilasi(i)=0
       enddo
      do i=1,maxnod
        nexti(i)=0
       enddo
c
c     initialise ifirsi to mark boundary channels
c
      ich=0
      do i=1,nbcha
        ich=inext(ich)
        ifirsi(ich)=-1
       enddo
c
      iprev2=0
      iprev3=0
      ilevel=0
      n2=0
      n3=nbcha
      ncon=1
      nint=0
      kin=2
      nnod=0
      ich=0
c
c     begin loop on levels
c
  10  do while (n3.gt.n2)
        ilevel=ilevel+1
        n1=n2+1
        n2=n3
        iprev1=iprev2+1
        iprev2=iprev3
c
c       begin loop on channels in level # ilevel
c
        do icoun=n1,n2
```

```
            ich=inext(ich)
c
            nif=0
c
c           look for previously stored nodes
c
            noldi=0
            iint=ifirsi(ich)
            if(ich.eq.297) call dum
            do while(iint.gt.0)
              noldi=noldi+1
              nif=nif+1
            if(nif.eq.296) call dum
            if(noldi.eq.2014) call dum
            if(nif.ne.noldi) call dum
              inod(nif)=iint
              iint=nexti(iint)
             enddo
c
c           mark channel as having been studied
c
            ifirsi(ich)=-2
c
c           regular channel
c
            if(ich.le.inch)then
              nloop=1
              ifol(1)=ifrach(ich)
              do j=1,4
                j3=j+3
                xy(j)=chan(ich,j3)
               enddo
             else
c
c             fracture intersection
c
              nloop=2
              k=ich-inch
              do j=1,2
                ifol(j)=inte2(j,k)
               enddo
              do j=1,8
                xy(j)=inte4(j,k)
               enddo
             endif
c
c           loop over fracture(s)
c
            do iloop=1,nloop
              ifold=ifol(iloop)
              x1=xy(4*iloop-3)
              y1=xy(4*iloop-2)
              x2=xy(4*iloop-1)
              y2=xy(4*iloop)
              xx=x1-x2
              yy=y1-y2
c
              jch1=ichan(1,ifold)-1
              nch=ichan(2,ifold)
              jint=inkey(ifold,2)-1
              nint1=inkey(ifold,1)
              ntot=nch+nint1
c
c             begin loop over channels not previously encountered
```

```
c
            do 30 icount=1,ntot
c
c             regular channel
c
              if(icount.le.nch)then
                jch=jch1+icount
                if(ifirsi(jch).eq.-2)goto 30
                xn1=chan(jch,4)
                yn1=chan(jch,5)
                xn2=chan(jch,6)
                yn2=chan(jch,7)
               else
c
c               fracture intersection
c
   20           jint=jint+1
                k=inte3(jint)
                if(k.eq.0)goto 20
                jch=inch+k
                if(ifirsi(jch).eq.-2)goto 30
                jfold=inte2(1,k)
                iplus=0
                if(jfold.ne.ifold)iplus=4
                xn1=inte4(iplus+1,k)
                yn1=inte4(iplus+2,k)
                xn2=inte4(iplus+3,k)
                yn2=inte4(iplus+4,k)
               endif
              xn=xn1-xn2
              yn=yn1-yn2
c
              a1=xn*(y1-yn2)-yn*(x1-xn2)
              a2=xn*(y2-yn2)-yn*(x2-xn2)
c
c             channels are on the same line
c
              if(abs(a1).le.stoler.and.abs(a2).le.stoler)then
                if(abs(xn).ge.toler)then
                  t1=(xn1-x1)/xn
                  t2=(xn1-x2)/xn
                 elseif(abs(yn).ge.toler)then
                  t1=(yn1-y1)/yn
                  t2=(yn1-y2)/yn
                 else
                  goto 30
                 endif
                if(t1.ge.0..and.t1.le.1.)then
                  tn=t1
                  t=0.
                 elseif(t2.ge.0..and.t2.le.1.)then
                  tn=t2
                  t=1.
                 else
                  goto 30
                 endif
               else
c
c               channels are not on the same line
c
                if(a1*a2.gt.0.)goto 30
                b2=(xn1-x2)*(y1-y2)-(x1-x2)*(yn1-y2)
                tn=b2/(a1-a2)
                if(tn.lt.0..or.tn.gt.1.)goto 30
```

```
                t =a1/(a1-a2)
c
c               Store channel/channel intersection
c
                nint=nint+1
                iprev3=iprev3+1
                nif=nif+1
                inod(nif)=nint
c
c               increment array nexti to keep track of intersections
c               on channel jch, and array inext for next level if
c               the channel had not been encountered previously.
c
                if(ifirsi(jch).le.0)then
                  if(ifirsi(jch).eq.0)then
                    n3=n3+1
                    inext(ilast)=jch
                    ilast=jch
                   endif
                  ifirsi(jch)=nint
                 else
                  jl=ilasi(jch)
                  nexti(jl)=nint
                 endif
                ilasi(jch)=nint
                tnod(1,nint)=t
                tnod(2,nint)=tn
                jnod(1,nint)=ich
                jnod(2,nint)=jch
c
               endif
c
c              end loop on new channels
c
   30         continue
c
c            end loop on fracture(s)
c
            enddo
c
c           if maximum number of intersections is exceeded, stop
c
            if (nint.gt.maxnod)stop 'too many channel intersections'
c
c           set elements array
c
            nodk(1,ich)=nnod+1
            nodk(2,ich)=nif
            if(nif.ne.0)then
              do i=1,nif
                nnod=nnod+1
                knod(nnod)=inod(i)
               enddo
             endif
c
c           end do loop on channels in level # ilevel
c
          enddo
c
c         end do loop on levels
c
        enddo
c
c     if ikeep is 2, inject any channel not connected to the
```

```
c     boundaries in next level, and go back to loop over levels
c
      if (ikeep.eq.2.and.n3.lt.nchan)then
        do jch=ncon,nchan
          if(ifirsi(jch).ne.-2)then
            inext(ich)=jch
            n3=n3+1
            ilast=jch
            ncon=jch+1
            goto 10
           endif
         enddo
       endif
      nchan=n3
      nnod=nnod/2
c
      return
c
      end
```

```
c
c
      subroutine limit
c
c     ****************************************************************
c
c     this subroutine truncates a channel if one or both of its
c     endpoints fall outside of the fracture on which it was generated.
c
c     if the channel is truncated, the coordinates of the endpoints
c     and the length of the channel  are recalculated.
c
c     if the channel is truncated at a fracture boundary intersection,
c     it is flagged as a boundary channel.
c
c     nchan is the number of channels.
c
c     some components of chan may be recalculated in this subroutine
c       chan(i,2) = length,
c       chan(i,4) = x1, x coordinate of endpoint 1,
c       chan(i,5) = y1, y coordinate of endpoint 1,
c       chan(i,6) = x2, x coordinate of endpoint 2,
c       chan(i,7) = y2, y coordinate of endpoint 2.
c
c       next(i) = where the next boundary channel sits
c
#include "change.cmn"
c
      dimension rotf(3,3)
      dimension tbound(200),tchan(200),ibound(200),iend(200)
      byte keep(200)
c
      ilast=0
      nbcha=0
      kch=0
      nhol1=nhole+nbhol+1      !jep 1jul88 - for square fracture simulation
      toler=1./(10.**itole)
      stoler=toler*toler
c
      do i=1,maxhed
        do j=1,2
          iboun(j,i)=0
         enddo
       enddo
      do i=1,maxhed
        do j=1,2
          tboun(j,i)=0.
         enddo
       enddo
c
c     compute upper limit inch of the number of channels in the flow
c     region.  The fracture intersections will be recorded by numbers
c     above inch.
c
      inch=0
      do ifrac=1,nfrac
        ifold=iref(ifrac)
        inch=inch+ichan(2,ifold)
       enddo
c
c     loop is over old fracture numbers to enable compression of
c     the list of channels.
c
      do ifold=1,nfracg
```

```
      ifrac=inew(ifold)
      if (ifrac.ne.0)then
        rad=frac(3,ifold)
c
c     compute matrix rotf for rotation from global to local system.
c
        call ROFRAC(ifold,rotf,zplane,dk)
        if(ifrac.le.nfracb)then
          k1=ibkey(ifold,2)
          k3=k1+ibkey(ifold,1)-1
c
c         rotate fracture boundary intersections from global to
c         local coordinates
c
          call rfrcglo(18,maxint+1,inte1,k1,k3,rotf,2)
         endif
c
        ich=ichan(1,ifold)-1
        ichan(1,ifold)=kch+1
        ncount=ichan(2,ifold)
c
        do 10 icount=1,ncount
          ich=ich+1
          alen=chan(ich,2)
          x1=chan(ich,4)
          y1=chan(ich,5)
          x2=chan(ich,6)
          y2=chan(ich,7)
c
c         truncate channel at boundary of fracture disc
c
          a=alen*alen
          b=x1*(x2-x1)+y1*(y2-y1)
          c=x1*x1+y1*y1-rad*rad
          delt=b*b-a*c
          if(delt.le.0.)stop 'channel outside its disc'
          delt=sqrt(delt)
          t1=(-b-delt)/a
          if(t1.ge.1.)stop 'channel outside its disc'
          t2=(-b+delt)/a
          if(t2.le.0.)stop 'channel outside its disc'
          if(t1.gt.0.)then
            x1=x1+t1*(x2-x1)
            y1=y1+t1*(y2-y1)
            t2=(t2-t1)/(1.-t1)
            alen=alen*(1.-t1)
           endif
          if(t2.lt.1.)then
            x2=x1+t2*(x2-x1)
            y2=y1+t2*(y2-y1)
            alen=alen*t2
           endif
c
c         initialize keep to
c         one for the parallelepipeds.
c
          do ihole=1,nhol1
            keep(ihole)=1
           enddo
c
c         record first endpoint of channel as first intersection
c
          int=1
          tchan(int)=0
```

```
            iend(int)=0
            ibound(int)=0
c
            if(ifrac.le.nfracb)then
c
c           loop over boundary intersections on fracture ifold
c           to record significant points.
c
              ihold=1
              do k=k1,k3
                xb1=inte1(1,k)-dk
                yb1=inte1(2,k)
                xb2=inte1(4,k)-dk
                yb2=inte1(5,k)
                isid=-inte2(2,k)
                ihole=(isid+5)/6
                if(ihole.ge.2.and.ihole.ne.ihold)keep(ihole)=0
                ihold=ihole
                xb=xb1-xb2
                yb=yb1-yb2
c
                a1=xb*(y1-yb2)-yb*(x1-xb2)
                a2=xb*(y2-yb2)-yb*(x2-xb2)
c
c               reset keep according position of first endpoint of channel
c
                if(a1.gt.0..and.ihole.ge.2)then
                  keep(ihole)=1
                 elseif(a1.le.0..and.ihole.eq.1)then
                  keep(ihole)=0
                 endif
c
                if(abs(a1-a2).gt.stoler)then
c
c                 channel and boundary line are not parallel
c
                  b2=(xb1-x2)*(y1-y2)-(x1-x2)*(yb1-y2)
                  tb=b2/(a1-a2)
                  ltoler=toler/alen
                  t=a1/(a1-a2)
c
                  if(tb.ge.ltoler.and.tb.le.1.-ltoler.
                     and.t.gt.0..and.t.lt.1.)then
c
c                   channel line cuts boundary between its endpoints
c
                    int=int+1
                    tchan(int)=t
                    tbound(int)=tb
                    ibound(int)=k
                    if(a1.le.0.)then
c
c                     'delete to exists' intersection
c
                      iend(int)=ihole
                     else
c
c                     'exists to delete' intersection
c
                      iend(int)=-ihole
                     endif
                   endif
                 endif
               enddo
```

```
            endif
c
c           sort intersections by ascending tchan's
c           don't sort intersection # 1, since we know it is the
c           first channel endpoint.
c
            i2=int
            j1=3
            do while (i2.ge.j1)
              j2=i2
              i0=1
              i1=0
              i2=0
              do 40 j=j1,j2
                if (tchan(j-1).le.tchan(j)) go to 40
                t=tchan(j-1)
                tchan(j-1)=tchan(j)
                tchan(j)=t
                t=tbound(j-1)
                tbound(j-1)=tbound(j)
                tbound(j)=t
                in=iend(j-1)
                iend(j-1)=iend(j)
                iend(j)=in
                in=ibound(j-1)
                ibound(j-1)=ibound(j)
                ibound(j)=in
                i2=j-1
                i1=i1+i0*i2
                i0=0
   40           continue
              j1=max0(i1,2)
            enddo
c
c           record second endpoint of channel as last intersection
c
            int=int+1
            tchan(int)=1.
            ibound(int)=0
            nintch=int
c
c           loop over segments to find which ones exist
c
            do int=1,nintch-1
              ihole=iend(int)
              if(ihole.gt.0)then
                keep(ihole)=1
              elseif(ihole.lt.0)then
                keep(-ihole)=0
              endif
              iexist=1
              do ihole=1,nhol1
                iexist=iexist*keep(ihole)
              enddo
              if(iexist.gt.0)then
                al=alen*(tchan(int+1)-tchan(int))
                if(al.ge.toler)then
c
c                 segment is long enough to be considered
c
                  if(ibound(int)+ibound(int+1).ne.0)then
c
c                 one or two endpoints are on boundary
c
```

```
                    nbcha=nbcha+1
                    inext(ilast)=kch+1
                    ilast=kch+1
                    iboun(1,nbcha)=ibound(int)
                    tboun(1,nbcha)=tbound(int)
                    iboun(2,nbcha)=ibound(int+1)
                    tboun(2,nbcha)=tbound(int+1)
                   endif
c
c              if there is no room in array chan, call
c              subroutine move to make some room.
c
                  if(kch.ge.ich)
                    call move(ich,ifold+1,ncount-icount)
c
                  kch=kch+1
                  chan(kch,1)=chan(ich,1)
                  chan(kch,2)=a1
                  chan(kch,3)=chan(ich,3)
                  chan(kch,4)=x1+tchan(int)*(x2-x1)
                  chan(kch,5)=y1+tchan(int)*(y2-y1)
                  chan(kch,6)=x1+tchan(int+1)*(x2-x1)
                  chan(kch,7)=y1+tchan(int+1)*(y2-y1)
                  chan(kch,8)=chan(ich,8)
                  chan(kch,9)=chan(ich,9)
                  chan(kch,10)=chan(ich,10)
                  ifrach(kch)=ifold
                 endif
               endif
             enddo
 10        continue
c
          if(ifrac.le.nfracb)then
            call rfrcglo(18,maxint+1,inte1,k1,k3,rotf,1)
            ichan(2,ifold)=kch+1-ichan(1,ifold)
           endif
c
c       rotate internal fracture intersections to local coordinates
c       and store in array inte4.
c
          ni=inkey(ifold,1)
          if(ni.gt.0)then
            ks=inkey(ifold,2)-1
            do n=1,ni
 20           ks=ks+1
              k=inte3(ks)
              if(k.eq.0)goto 20
              k=k+inst
              inte3(ks)=k
              call rfrcglo(18,maxint+1,inte1,k,k,rotf,2)
c
              if(ifold.eq.inte2(1,k))then
c
c             check for fracture intersection with endpoints on
c             boundary of flow region
c
                ibb=0
                do is=1,2
                  if(inside(is,k).ne.0)then
                    if(ibb.eq.0)then
                      ich=inch+k
                      inext(ilast)=ich
                      ilast=ich
                      nbcha=nbcha+1
```

```
                    ibb=1
                   endif
                  iboun(is,nbcha)=k
                  tboun(is,nbcha)=float(is-1)
                 endif
                enddo
c
                inte4(1,k)=inte1(1,k)-dk
                inte4(2,k)=inte1(2,k)
                inte4(3,k)=inte1(4,k)-dk
                inte4(4,k)=inte1(5,k)
               else
                inte4(5,k)=inte1(1,k)-dk
                inte4(6,k)=inte1(2,k)
                inte4(7,k)=inte1(4,k)-dk
                inte4(8,k)=inte1(5,k)
               endif
              call rfrcglo(18,maxint+1,inte1,k,k,rotf,1)
             enddo
            endif
           endif
          enddo
      nchan=kch
      if(nbcha.gt.maxhed)stop 'too many boundary channels'
c
      return
      end
```

```
c
c
      subroutine lognod (a,n,dseed,ev,sd)
c
c *** this subroutine generates random variables distributed
c *** lognormally with expected value ev and standard distribution sd.
c
c *** step 1.  sn = sum of 25 random variables distributed uniformly
c ***          in (0,1),
c *** step 2.  sn = (sn-12.5)*sqrt(.48) is distributed normally
c ***          with mean = 0 and standard deviation = 1,
c *** step 3.  exp(sd*sn+ev) is distributed lognormally with mean =
c ***          exp(ev)*exp(sd*sd/2) and standard deviation =
c ***          exp(sd*sd+2*ev)*(exp(sd*sd)-1).
c
      dimension a(n)
      data s48/0./
c
      if (s48.eq.0) s48=sqrt(.48)
      a2evsd=alog(ev*ev+sd*sd)
      aev=alog(ev)
c
      evn=2.*aev-0.5*a2evsd
      sdn=sqrt(a2evsd-2.*aev)
c
      do 120 i=1,n
        sn=0.
        do 110 j=1,25
          sn=sn+ggubfs(dseed)
  110     continue
        sn=s48*(sn-12.5)
  120   a(i)=exp(sdn*sn+evn)
c
      return
      end
```

```
c
      subroutine move(ich,ifold1,nleft)
c
c     moves the information in the array chan and in the pointer
c     to it: ichan(1,nfold), for all fractures with number greater than
c     ifold.  This way, some room is made for newly created boundary
c     channels due to the splitting of a channel at an inner boundary.
c
#include "change.cmn"
c
      if (nchan.eq.maxcha)then
        stop  'too many channels created at inner boundaries'
       else
        nmove=jmin0((nchan-ich)/20+10,maxcha-nchan)
        nchan=nchan+nmove
       endif
c
c     move channels on other fractures
c
      do jfold=nfracg,ifold1,-1
        ifrac=inew(jfold)
        if(ifrac.ne.0)then
          n1=ichan(1,jfold)
          n2=n1+ichan(2,jfold)-1
          ichan(1,jfold)=n1+nmove
          do jch=n2,n1,-1
            do i=1,10
              chan(jch+nmove,i)=chan(jch,i)
             enddo
           enddo
         endif
       enddo
c
c     move channels left on current fracture
c
      nlast=ich+nleft
      do jch=nlast,ich,-1
        do i=1,10
          chan(jch+nmove,i)=chan(jch,i)
         enddo
       enddo
      ich=ich+nmove
      return
      end
```

```
c
      subroutine normad (a,n,dseed,ev,sd)
c
c *** this subroutine generates random variables distributed
c *** normally with expected value ev and standard distribution sd.
c
c *** step 1.  sn = sum of 25 random variables distributed uniformly
c ***          in (0,1),
c *** step 2.  sn = (sn-12.5)*sqrt(.48) is distributed normally
c ***          with mean = 0 and standard deviation = 1,
c *** step 3.  sd*sn+ev is distributed normally with mean ev and
c ***          standard deviation sd.
c
      dimension a(n)
      double precision dseed
      data s48/0./
c
      if (s48.eq.0) s48=sqrt(.48)
      do 120 i=1,n
        sn=0.
        do 110 j=1,25
  110     sn=sn+ggubfs(dseed)
        sn=s48*(sn-12.5)
  120   a(i)=sd*sn+ev
c
      return
      end
```

```
      subroutine normdl (a,n,dseed,ycept,sd,slope,b,ichar)
c
c *** this subroutine generates random variables distributed
c *** normally with expected value ev and standard distribution sd.
c *** where ev is proportional to the logarithm of another
c *** parameter or the parameter itself
c
c *** step 1.  sn = sum of 25 random variables distributed uniformly
c ***          in (0,1),
c *** step 2.  sn = (sn-12.5)*sqrt(.48) is distributed normally
c ***          with mean = 0 and standard deviation = 1,
c *** step 3.  sd*sn+ev is distributed normally with mean ev and
c ***          standard deviation sd. where ev is proportional
c ***          to a parameter or the log of the parameter.
c *** step 4.  the value of sd*sn+ev is set so that it is never
c ***          less than a minimum value
c
      dimension a(n),b(n)
      double precision dseed
      data emin/1.e-8/
      data s48/0./
c
      if (s48.eq.0) s48=sqrt(.48)
      do 120 i=1,n
        sn=0.
        do 110 j=1,25
  110     sn=sn+ggubfs(dseed)
        sn=s48*(sn-12.5)
        f1=b(i)
        if (ichar.eq.5)then
          ev=ycept+slope*f1
         else
          ev=ycept+slope*alog10(f1)
         endif
        a(i)=sd*sn+ev
        if (a(i).lt.emin) a(i)=emin
  120   continue
c
      return
      end
```

```
      subroutine oriest(k,n1,n2,ev,sd)
c
c     ***** NOTICE ******:
c       This subroutine is stator.
c     stator was taken from [fanay.program.fmg] on 3/7/86
c     to replace oriest (which did not calculate the
c     correct values for the sd and ev of the orientation.
c                                                -KTN
c
c     basic statistics for orientation distributions.
c
c       this subroutine may not be accurate if the
c       range of angles in the distribution is more
c       than 90 degrees wide.
c
c
#include "change.cmn"
c
      pi=3.141592
      if(maxcha.eq.0)then
        ev=0.
        sd=0.
        return
      endif
      amsin=0.
      amcos=0.
      do 10 i=n1,n2
      chan(i,k)=chan(i,k)*pi/180.
      asi=sin(chan(i,k))
      aco=cos(chan(i,k))
      s1=amsin+asi
      c1=amcos+aco
      s2=amsin-asi
      c2=amcos-aco
      v1=s1**2+c1**2
      v2=s2**2+c2**2
      if(v1.ge.v2)then
        amsin=s1
        amcos=c1
       else
        amsin=s2
        amcos=c2
      endif
   10 continue
      if(abs(amcos).le.1.e-10)then
        ev=pi/2.
       else
        ev=atan(amsin/amcos)
      endif
      sd=0.
      do 20 i=n1,n2
      alph=chan(i,k)
      dif=abs(ev-alph)
   18 continue
      if(dif.gt.pi/2.)then
        dif=abs(pi-dif)
        goto18
      endif
   20 sd=sd+dif**2
      sd=sd/(n2-n1+1)
      sd=sqrt(sd)
       do 30,i=n1,n2
   30  chan(i,k)=chan(i,k)*180./pi
       ev=ev*180./pi
```

```
 sd=sd*180./pi
return
end
```

```
      subroutine randxy (x,y,n,dseed,rad,itole)
c
c *** this subroutine generates random channel  centers
c
      dimension x(n),y(n)
      double precision dseed
c
      toler=10.**itole
      diam=2.*rad*toler
      srad=rad*rad
c
      do i=1,n
 10     x(i)=float(int(diam*ggubfs(dseed)))/toler-rad
        y(i)=float(int(diam*ggubfs(dseed)))/toler-rad
        dist=x(i)*x(i)+y(i)*y(i)
        if(dist.gt.srad)goto 10
       enddo
c
      return
      end
```

```
      subroutine rchan
c
c     ******************************************************************
c
c     this subroutine reads in channel information
c     on the following channel characteristics -
c     orientation, length, aperture, and the coordinates of the
c     channel center.  The channel sets to be used for each fracture set
c     are also read.
c
c     the following variables or arrays from the main program are
c     used ---
c
c       nchst = number of channel sets,
c
c
c     the variables which control the read or generation are given
c     below ---
c
c       icent = 1 - generate channel centers on an evenly spaced grid
c               2 - generate channel center coordinates randomly
c
c       idens = 0 - input rlamb (density), compute nchan
c               1 - input nchan/fracture
c
c       ichar - read value for each of orientation, length and
c               aperture,
c             = 2 - set characteristic equal to a constant value,
c             = 3 - generate characteristics values according to a
c                   distribution,
c             = 4 - generate apertures correlated to log of length
c             = 5 - generate apertures correlated to length
c             = 6 - generate apertures with fracture aperture as mean
c
c       idist = 1 - normal distribution,
c               2 - lognormal distribution,
c               3 - exponential distribution.
c               4 - gamma distribution
c               5 - uniform distribution.
c
c     in addition, the following variables are read in ---
c
c       rlamb = no. of channels per square unit,
c
c       itole - number of decimal places in channel center coordinates
c
c       const - see ichar = 2,
c
c       ev - expected value of statistical distribution,
c
c       sd - standard deviation of statistical distribution.
c
c     the array ichsi is used to store the values of nchan, icent, and
c     ichar and idist for each set.  the array rchsi is used to store
c     the values of const, ev and sd for each set.
c
c     ******************************************************************
c
c
#include "change.cmn"
c
c *** zero information matrices
c
      do 20 i=1,nchst
```

```
          do 10 j=1,8
            ichsi(i,j)=0
            rchsi(i,j)=0.
   10       continue
          rchsi(i,9)=0.
   20     continue
c
        do iset=1,nsets
          read (1,90) (isetch(iset,i),i=1,10)
         enddo
c
        itrans = 0
        do 70 i=1,nchst
c
          read (1,100) icent,idens                    !jep 29sep88
          if(icent.lt.0) then
            itrans= 1
            icent = -icent
          end if
c
c         only options available now is generate at random
c         or evenly spaced centers (icent = 1)         jep 19jul88
c
          ichsi(i,1)=icent
          ichsi(i,9)=idens                            !jep 29sep88
c
          read (1,110) rlamb
          rchsi(i,10)=rlamb
c
          l=1
          m=-2
          do 60 k=1,3
            l=l+2
            m=m+3
c
            read (1,100) ichar
            ichsi(i,l)=ichar
c
c *** set channel characteristic to a constant
c
            if (ichar.lt.3) then
              read (1,110) const
              rchsi(i,m)=const
c
c *** read in code for statistical distribution and read in
c *** statistical parameters
c
             elseif (ichar.eq.3.or.ichar.eq.6) then
              read (1,120) idist,ev,sd
              ichsi(i,l+1)=idist
              rchsi(i,m+1)=ev
              rchsi(i,m+2)=sd
             else
              read (1,130) ycept,slope,sd
              ichsi(i,l+1)=1
              rchsi(i,m)=ycept
              rchsi(i,m+1)=slope
              rchsi(i,m+2)=sd
             endif
c
   60       continue
   70     continue
c
        return
```

```
c
   90 format(10x,10i5)
  100 format(2(10x,i5))
  110 format(10x,e10.4)
  120 format(10x,i5,15x,f10.4,10x,f10.4)
  130 format(3(10x,f10.4))
  140 format(5f10.4)
      end
```

```
c
      subroutine rfract (oray,odate,title2)
c
c *** This subroutine reads all primary fracture
c *** system data from unit 5.
c
#include "change.cmn"
      character*80 title2(2)
      character oray*19,odate*9
c
c
      open(unit=5,file='frac3d.dat',status='old',form=
     ''unformatted')
      read (5) oray,odate
      read (5) title2
      read (5) nsets,itole,ikeep
      read (5) ((k,j=1,8),i=1,nsets)
      read (5) ((x,j=1,16),i=1,nsets)
      read (5) nfrac,x
      if (nfrac.gt.maxfrc)stop 'too many fractures'
      do i=1,nfrac
        read (5) iold,(frac(j,i),j=1,12),
     '             isetfr(i),k,k
       enddo
      close(unit=5)
      return
      end
```

```
c
      SUBROUTINE RFRCGLO(isiz1,isiz2,xyz,k1,k2,rotf,inv)
c
c *** This subroutine transforms coordinates from the local
c *** fracture system to the global XYZ coordinate system
c *** or viceversa according to inv = 1 or 2, respectively,
c *** using the rotation matrix rotf.
c
      dimension xyz(isiz1,isiz2),rotf(3,3)
      dimension txyz(6),rot(3,3)
c
c
c *** Rotate from fracture coord. to global coord.
      if(inv.eq.1)then
        do i=1,3
          do j=1,3
            rot(i,j)=rotf(i,j)
          enddo
        enddo
       elseif(inv.eq.2)then
c
c ***    Rotate from global coord. to fracture coord.
c
         do i=1,3
           do j=1,3
             rot(i,j)=rotf(j,i)
           enddo
         enddo
       endif
c
      do k=k1,k2
        j=0
c
        do j1=1,2
          nj1=3*j1-3
          do i1=1,3
            j=j+1
            txyz(j)=0.0
            nj=nj1
            do i2=1,3
              nj=nj+1
              txyz(j)=txyz(j)+rot(i1,i2)*xyz(nj,k)
            enddo
          enddo
        enddo
c
        do n=1,6
          xyz(n,k)=txyz(n)
        enddo
c
       enddo
c
  100 return
c
      end
```

```
c
      subroutine rinter (nflow)
c
c *** read intersection  and fracture data from program FMG3D
c
c
#include "change.cmn"
c
      character file*12
c
      write(file,40)nflow
   40 format('change',i2.2,'.dat')
      open(unit=9,file=file,status='old',
     '     err=80)
c
      read(9,900) nfracg,nfrac,nfracb,nbpt,inst,ninb,
     '            rgene,nhole,nbhol,xmesh,ymesh,zmesh,rophi,rothe
      read(9,910)
      read(9,980) ((frcoef(i,j),i=1,4),j=1,6*(nhole+1))
      read(9,910)
      read(9,930) (iref(i),i=1,nfrac)
      read(9,910)
      read(9,920) ((frac(j,i),j=1,12),i=1,nfracg)
      read(9,910)
      read(9,940) ((ibkey(i,j),j=1,2),i=1,nfracg)
      read(9,910)
      read(9,940) ((inkey(i,j),j=1,2),i=1,nfracg)
      read(9,910)
      read(9,940) ((inside(j,i),j=1,2),i=1,nbpt)
      read(9,910)
      read(9,940) ((inte2(j,i),j=1,2),i=1,nbpt)
      read(9,950) inte3(1)
      read(9,930) (inte3(i),i=2,inte3(1))
      read(9,910)
      read(9,960) ((inte1(j,i),j=1,6),i=1,nbpt)
c
      close(unit=9)
c
c     initialize array inew
c
      do i=1,nfracg
        inew(i)=0
       enddo
      do i=1,nfrac
        iold=iref(i)
        inew(iold)=i
       enddo
      return
   80 stop
c
  900 format (2(10x,i5,15x,i5,15x,i5/),10x,f10.0,10x,i5,10x,i5/
     '        3(10x,f10.0)/2(10x,f10.0))
  910 format (1x)
  920 format (6(1x,e12.5))
  930 format (15i5)
  940 format (5(i5,i6,4x))
  950 format (9x,i6)
  960 format (6f10.4)
  980 format (4(1x,e12.5))
      end
```

```
c
      SUBROUTINE ROFRAC (ifold,rotf,zplane,dk)
c
c *** This subroutine computes the matrix of rotation,
c *** rotf, from the global-XYZ coordinate system to a
c *** system convenient to the plane of fracture ifold.
c *** The fracture plane and center are simplified to
c *** z=-d and (dk,0,-d), respectively.
c
      parameter (pi180=0.0174532925)
#include "change.cmn"
      dimension rotf(3,3)
c
c
c *** Transfer coefficients a,b,c,d
      do 100 i=1,3
  100 rotf(i,3)=frac(i+8,ifold)
      d=frac(12,ifold)
c
c *** Compute l1,l2,l3
      dk=0.
      do 110 i=1,3
      rotf(i,1)=frac(i+4,ifold)+rotf(i,3)*d
  110 dk=dk+rotf(i,1)**2
      dk=sqrt(dk)
      if ( dk .lt. 0.005 ) go to 140
c
      do 120 i=1,3
  120 rotf(i,1)=rotf(i,1)/dk
c
c *** Compute m1,m2,m3
      do 130 i=1,3
      j=i+2
      if (j.gt.3) j=mod(j,3)
      k=6-i-j
  130 rotf(i,2)=rotf(j,1)*rotf(k,3)-rotf(k,1)*rotf(j,3)
      go to 200
c
  140 dk=0.0
      phi=frac(1,ifold)*pi180
      theta=frac(2,ifold)*pi180
      rotf(1,1)= cos(phi)*cos(theta)
      rotf(2,1)=-cos(phi)*sin(theta)
      rotf(3,1)=-sin(phi)
      rotf(1,2)= sin(theta)
      rotf(2,2)= cos(theta)
      rotf(3,2)= 0.0
c
  200 zplane=-d
c
      return
c
      end
```

```
      subroutine unifod (a,n,dseed,center,range)
c
c     this subroutine generates random variables uniformly
c     distributed,with minimum center-range and maximum center+range.
c
      dimension a(n)
      double precision dseed
c
      ainf=center-range
      span=2*range
      do 10 i=1,n
        a(i)=ainf+ggubfs(dseed)*span
   10   continue
      return
      end
```

```
c
      subroutine wlines(nflow)
c
c     This subroutine writes the input file for the plotting program,
c     DIMES, if iplot = 2
c
#include "change.cmn"
c
      dimension txyz(6)
      character file*11
c
      write(file,40)nflow
   40 format('lines',i2.2,'.dat')
      open(unit=7,file=file)
c
c     write channels
c
      ich=0
      do i=1,nchan
        ich=inext(ich)
        if(ich.le.inch)then
          ifrac=ifrach(ich)
          jfrac=ifrac
          do j=1,6
            txyz(j)=chan(ich,j+3)
           enddo
         else
          iint=ich-inch
          ifrac=inte2(1,iint)
          jfrac=inte2(2,iint)
          do j=1,6
            txyz(j)=inte1(j,iint)
           enddo
         endif
        write (7,710) txyz,inew(ifrac),inew(jfrac)
c
       enddo
c
      close (unit=7)
c
      return
c
  710 format(6f10.4,2i5)
      end
```

```
c
      subroutine wrenum
     ' (nflow,iplot,iray,idate,title,oray,odate,title2)
c
c     This subroutine writes the input file for the next program,
c     RENUM
c
#include "change.cmn"
      common /qc/ visc,spgr,qc
c
      dimension inod(4000),tkut(4000)  !jep 1jul88 to prevent write-over
      dimension rotf(3,3)
      dimension xyz(3),txyz(6)
      character*80 title(2),title2(2)
      character*9 idate,odate
      character*19 iray,oray
      character file*11
c
      write(file,10)nflow
   10 format('renum',i2.2,'.dat')
c
      open (unit=4,file=file,
     1      carriagecontrol='list')
c
      write (4,410) iray,idate,iprnt,oray,odate,
     '              title(1),
     '              title2(1),
     '              nchan,nfracg,nfrac,nfracb,nbpt,inst,itole,
     '              rgene,rgene,rgene,
     '              xmesh,ymesh,zmesh,
     '              rophi,rothe,
     '              ndir,
     '              delphi,
     '              visc,spgr,
     '              nelem,nnodes,maxd,ikeep,iplot
c
c     rotate channel endpoints to global coordinates
c
      ich=0
      ifraco=0
      do i=1,nchan
        ich=inext(ich)
        if(ich.le.inch)then
          ifrac=ifrach(ich)
          if (ifrac.ne.ifraco)then
            call ROFRAC(ifrac,rotf,zplane,dk)
            ifraco=ifrac
           endif
          jk=0
          do j=1,2
            j2d=2*(j-1)
            xyz(1)=chan(ich,4+j2d)+dk
            xyz(2)=chan(ich,5+j2d)
            xyz(3)=zplane
c
            do k=1,3
              jk=jk+1
              txyz(jk)=0.0
              do l=1,3
                txyz(jk)=txyz(jk)+rotf(k,l)*xyz(l)
               enddo
             enddo
           enddo
          do j=1,6
```

```
            chan(ich,j+3)=txyz(j)
           enddo
          endif
         enddo
c
c     write nodes
c
c     boundary nodes
c
      ich=0
      ibnd=0
      do i=1,nbcha
        ich=inext(ich)
        do j=1,2
          ibn=iboun(j,i)
          if(ibn.ne.0)then
            ibnd=ibnd+1
            ifrac=inte2(1,ibn)
            if(ich.le.inch)then
              do k=1,3
                xyz(k)=chan(ich,k+3)+float(j-1)*
     ,                 (chan(ich,k+6)-chan(ich,k+3))
              enddo
             else
              do k=1,3
                xyz(k)=intel(k,ibn)+float(j-1)*
     ,                 (intel(k+3,ibn)-intel(k,ibn))
              enddo
             endif
            iside=inside(j,ibn)
            write (4,450)ibnd,ib(j,i),iside,ifrac,xyz,
     ,                   (hd(j+2*k,i),k=0,maxd-1)
           endif
         enddo
        enddo
      nbnd=ibnd
      nends=ibnd
c
c     if ikeep is not 0, write fracture endpoints nodes.
c
      ibb=0
      iside=0
      if(ikeep.ne.0)then
        ich=0
        nend=ibnd
c
c       non boundary nodes on boundary channels
c
        do i=1,nbcha
          ich=inext(ich)
          do j=1,2
            ibn=iboun(j,i)
            if(ibn.eq.0)then
              nend=nend+1
              if(ich.le.inch)then
                ifrac=ifrach(ich)
                do k=1,3
                  xyz(k)=chan(ich,k+3)+float(j-1)*
     ,                   (chan(ich,k+6)-chan(ich,k+3))
                enddo
               else
                iint=ich-inch
                ifrac=inte2(1,iint)
                do k=1,3
```

```
                xyz(k)=intel(k,iint)+float(j-1)*
     '                  (intel(k+3,iint)-intel(k,iint))
               enddo
             endif
            write (4,450)nend,ibb,iside,ifrac,xyz
           endif
         enddo
       enddo
c
c      internal channels
c
       do i=nbcha+1,nchan
         ich=inext(ich)
         do j=1,2
           nend=nend+1
           ifrac=inte2(1,ibn)
           if(ich.le.inch)then
             ifrac=ifrach(ich)
             do k=1,3
               xyz(k)=chan(ich,k+3)+float(j-1)*
     '                 (chan(ich,k+6)-chan(ich,k+3))
              enddo
            else
             iint=ich-inch
             ifrac=inte2(1,iint)
             do k=1,3
               xyz(k)=intel(k,iint)+float(j-1)*
     '                 (intel(k+3,iint)-intel(k,iint))
              enddo
            endif
           write (4,450)nend,ibb,iside,ifrac,xyz
          enddo
       enddo
c
         nends=nend
        endif
c
c      internal nodes
c
       do ind=1,nnod
         ich=jnod(1,ind)
         t=tnod(1,ind)
         if(ich.le.inch)then
           ifrac=ifrach(ich)
           do k=1,3
             xyz(k)=chan(ich,k+3)+t*
     '        (chan(ich,k+6)-chan(ich,k+3))
            enddo
          else
           iint=ich-inch
           ifrac=inte2(1,iint)
           do k=1,3
             xyz(k)=intel(k,iint)+t*
     '               (intel(k+3,iint)-intel(k,iint))
            enddo
          endif
         nd=ind+nends
         write (4,450)nd,ibb,iside,ifrac,xyz
        enddo
c
c      write elements
c
       toler=10.**-itole
       nseg=0
```

```
      ich=0
      ibnd=0
      nend=nbnd
      do i=1,nchan
        ich=inext(ich)
        nif=nodk(2,ich)
        if(ich.le.inch)then
          alen=chan(ich,2)
          transm=chan(ich,3)
          if(itrans.eq.0) transm=qc*transm**3
          ifrac=ifrach(ich)
          jfrac=ifrac
         else
          iint=ich-inch
          xx=inte4(3,iint)-inte4(1,iint)
          yy=inte4(4,iint)-inte4(2,iint)
          alen=sqrt(xx*xx+yy*yy)
          ifrac=inte2(1,iint)
          jfrac=inte2(2,iint)
          aper1=frac(4,ifrac)
          aper2=frac(4,jfrac)
          aper=amax1(aper1,aper2)
          transm = aper
          if(itrans.eq.0) transm=qc*aper**3
         endif
        nod1=nodk(1,ich)
        nod2=nod1+nif-1
c
        in=0
        if(i.le.nbcha)then
          do iend=1,2
            if(iboun(iend,i).ne.0)then
              ibnd=ibnd+1
              ibn=nif*(iend-1)+1
              tkut(ibn)=float(iend-1)
              inod(ibn)=ibnd
              ibn=ibn*(2-iend)
              in=in+2-iend
              nif=nif+1
             elseif(ikeep.ne.0)then
              nend=nend+1
              nnend=nif*(iend-1)+1
              tkut(nnend)=float(iend-1)
              inod(nnend)=nend
              nnend=nnend*(2-iend)
              in=in+2-iend
              nif=nif+1
             endif
           enddo
         elseif(ikeep.ne.0)then
          nif=nif+2
          tkut(1)=0.
          tkut(nif)=1.
          inod(1)=nend+1
          nend=nend+2
          inod(nif)=nend
          in=1
         endif
c
        do ient=nod1,nod2
          in=in+1
          nod=knod(ient)
          inod(in)=nod+nends
          if(jnod(1,nod).eq.ich)then
```

```
            tkut(in)=tnod(1,nod)
           else
            tkut(in)=tnod(2,nod)
           endif
         enddo
c
c         sort elements by ascending tkut's
c
        i2=nif
        j1=2
        do while (i2.ge.j1)
          j2=i2
          i0=1
          i1=0
          i2=0
          do 40 j=j1,j2
            if (tkut(j-1).le.tkut(j)) go to 40
            t=tkut(j-1)
            tkut(j-1)=tkut(j)
            tkut(j)=t
            in=inod(j-1)
            inod(j-1)=inod(j)
            inod(j)=in
            i2=j-1
            i1=i1+i0*i2
            i0=0
   40       continue
          j1=max0(i1,2)
         enddo

c
        do iseg=1,nif-1
          nseg=nseg+1
          al=alen*(tkut(iseg+1)-tkut(iseg))
          if (al.le.toler)al=0.
          write(4,460)nseg,inod(iseg),inod(iseg+1),transm,al,
     ,                inew(ifrac),inew(jfrac)
         enddo
c
c       if ikeep is 0, change channel endpoints to real endpoints
c
        if(ikeep.eq.0)then
          if(ich.le.inch)then
            jk=0
            do j=1,nif,nif-1
              do k=1,3
                jk=jk+1
                txyz(jk)=chan(ich,k+3)+tkut(j)*
     ,          (chan(ich,k+6)-chan(ich,k+3))
               enddo
             enddo
            do j=1,6
              chan(ich,j+3)=txyz(j)
             enddo
           else
            iint=ich-inch
            jk=0
            do j=1,nif,nif-1
              do k=1,3
                jk=jk+1
                txyz(jk)=intel(k,iint)+tkut(j)*
     ,          (intel(k+3,iint)-intel(k,iint))
               enddo
             enddo
```

```
            do j=1,6
              intel(j,iint)=txyz(j)
            enddo
          endif
        endif
c
      enddo
      close (unit=4)
c
      return
c
  410 format(a19,' - ',a9,i1,9x,a19,' - ',a9,10x/2(a/),
     '      10hnchannels-,i5,5x,10hdisc mesh-,6i5/
     '     10hxgene- - -,f10.4,10hygene- - -,f10.4,10hzgene- - -,f10.4/
     '    10hxmesh- - -,f10.4,10hymesh- - -,f10.4,10hzmesh- - -,f10.4/
     '   10hphi- - - -,f10.4,10htheta- - -,f10.4/
     '  10hflow dir.-,3i10 /
     ' 10hheads- - -,3f10.3/
     '10hviscosity-,f10.4,10hspec.grav.,f10.4/
     '10hnelements-,i6,5x,10hnnodes - -,i6,5x,10hnboundary-,i6,5x/
     '10htrunc.code,i5,5x,10hplot code-,i5)
  450 format(4i5,6f10.4)
  460 format(3i5,5x,1p,2e10.3,5x,0p,2i5)
      end
```

Appendix C

Sample Input Files

File CHANGE.Inp

```
ICONT* * *    2      IPLOT* * *    2      IUNITS * *    1
IRANF* * *    1      RSEED* * *500311543.d0
MULTI-WELL ANALYSIS OF CHANNEL FLOW
JAN 19, 1988
NCHST* * *    3
ISETCHset1    1    2
ISETCHset2    1    3
ISETCHset3    2    3
ICENT*set1    2
RLAMB* * *  .30
ICHAR orie    3
IDIST* * *    1      EV * * * *45.0       SD * * * *5.000
ICHAR leng    3
IDIST* * *    2      EV * * * *3.         SD * * * *1.000
ICHAR aper    3
IDIST* * *    2      EV * * * *.001       SD * * * *0.0005
ICENT set2    2
RLAMB* * *  .50
ICHAR orie    3
IDIST* * *    1      EV * * * *-45.       SD * * * *25.000
ICHAR leng    3
IDIST* * *    2      EV * * * *5.         SD * * * *2.000
ICHAR aper    3
IDIST* * *    2      EV * * * *.001       SD * * * *0.001
ICENT set3    2
RLAMB* * *  .40
ICHAR orie    3
IDIST* * *    1      EV * * * *0.         SD * * * *15.000
ICHAR leng    3
IDIST* * *    2      EV * * * *2.         SD * * * *0.500
ICHAR aper    3
IDIST* * *    2      EV * * * *.0005      SD * * * *0.0001
ibcc  flow    2
ndir * * *    1
seth * * *    0.        0.        0.        0.        0.        0.
ibcc hole1    3
ndir * * *    1
seth * * * 0.0001
ibcc hole2    3
ndir * * *    1
seth * * *    0.
ibcc hole3    3
ndir * * *    1
seth * * *    0.
ibcc hole4    3
ndir * * *    1
seth * * *    0.
ibcc hole5    3
ndir * * *    1
seth * * *    0.
```

Appendix C (continued)

Sample Input Files

File CHANGE01.DAT

```
  nfracg   335          nfrac   149          nfracb   127
    nbpt   471           inst   224            ninb   217
   rgene    20.0000     nhole     5
   xmesh    20.0000     ymesh    20.0000     zmesh    20.0000
     phi     0.0000     theta     0.0000
(only the first five lines of each array are shown below)
 frcoef
  0.10000E+01  0.00000E+00  0.00000E+00 -0.10000E+02
  0.00000E+00  0.00000E+00  0.10000E+01 -0.10000E+02
  0.10000E+01  0.00000E+00  0.00000E+00  0.10000E+02
  0.00000E+00  0.00000E+00  0.10000E+01  0.10000E+02
  0.00000E+00  0.10000E+01  0.00000E+00 -0.10000E+02
(31 more lines)
 iref
    4    5    6   11   13   16   17   18   19   20   22   23   25   27   29
   31   32   36   40   42   43   45   46   47   48   49   50   52   53   55
   60   61   63   64   66   70   75   77   81   82   83   85   87   90   92
   95   96   99  100  108  109  110  113  116  123  124  126  134  135  143
  149  156  158  166  169  173  175  181  185  192  193  197  202  204  205
(5 more lines)
 frac
  0.11856E+02  0.18055E+03  0.40000E+01  0.10000E-03 -0.27000E+01 -0.16500E+01
  0.30900E+01  0.50265E+02 -0.20544E+00  0.19546E-02  0.97867E+00 -0.35756E+01
  0.18414E+02  0.18193E+03  0.40000E+01  0.10000E-03  0.16900E+02 -0.60001E-01
  0.72000E+00  0.47317E+02 -0.31570E+00  0.10631E-01  0.94880E+00  0.46528E+01
  0.14317E+02  0.16923E+03  0.40000E+01  0.10000E-03 -0.12360E+02  0.35800E+01
(665 more lines)
 ibkey
    0    0         0    0         0    0         2    1         2    3
    2    5         0    0         0    0         0    0         0    0
    3    7         0    0         2   10         0    0         0    0
    2   12         2   14         2   16         4   18         1   22
    0    0         2   23         2   25         0    0         2   27
(62 more lines)
 inkey
    6  389         0    0         0    0         0    2         2    2
    2    4         0    0         0    0         0    0         0    0
    1    6         0    0         1    7         0    0         0    0
    0    8         0    8         1    8         5    9         4   14
    0    0         0   18         3   18         0    0         2   21
(62 more lines)
 inside
    3    3         6    6         1    1         6    6         3    3
    6    6         6    6        25   25        30   30         1    1
    5    5         3    3         6    6         3    3         6    6
    1    1         6    6         7    7         9    9        11   11
   12   12         1    1         1    1         6    6         6    6
(90 more lines)
 inte2
    4   -3         4   -6         5   -1         5   -6         6   -3
    6   -6        11   -6        11  -25        11  -30        13   -1
   13   -5        16   -3        16   -6        17   -3        17   -6
   18   -1        18   -6        19   -7        19   -9        19  -11
   19  -12        20   -1        22   -1        22   -6        23   -6
(90 more lines)
 inte3      495 entries
    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30
   31   32   33   34   35   36   20   37   38   39   40   41   42   43   44
   45   46   47   48   49   50   51   52   53   54   55   56   57   58   59
   60   61   62   63   64   65   66   67   68   69   70   71   72   73   74
(28 more lines)
 inte1
  -10.0000  -10.0000    2.9279  -10.0000   -9.7807    2.9216
```

```
   -9.9241  -10.0000    2.9024  -10.0000  -10.0000    2.9279
   10.0000   -3.8831   -6.7533   10.0000  -10.0000   -7.0044
   10.0000  -10.0000   -7.0044    6.5938  -10.0000   -8.1351
  -10.0000  -10.0000   -7.2380  -10.0000   -6.6615   -7.2171
(466 more lines)
```

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