LBL--24915

DE90 008065

**FMG, RENUM, LINEL, ELLFMG, ELLP and DIMES: Chain of
Programs for Calculating and Analyzing Fluid Flow
through Two-Dimensional Fracture Networks.
Users Manuals and Listings**

Daniel Billaux, John Peterson, Sorin Bodea and Jane Long

**Earth Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, California 94720**

September 30, 1989

**This work was supported by the Repository and Technology Program of the Office of
Civilian Radioactive Waste Management of the U. S. Department of Energy under
Contract No. DE-AC03-76SF00098.**

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Table of Contents

List of Tables	v
Acknowledgement	vii
1.0 Introduction	1
2.0 Program FMG	3
3.0 Program RENUM	13
4.0 Program LINEL	19
5.0 Program ELLFMG	25
Appendix A: FMG - Program Organization and Arrays	29
Appendix B: FMG - Program Listing	49
Appendix C: RENUM - Program Organization and Arrays	95
Appendix D: RENUM - Program Listing	105
Appendix E: LINEL - Program Organization and Arrays	121
Appendix F: LINEL - Program Listing	131
Appendix G: ELLFMG - Program Organization and Arrays	149
Appendix H: ELLFMG - Program Listing	155

List of Tables

Table	Title	Page
2-1	FMG - Input Variables and Formats	5
2-2	FMG - Description of Input Variables	7
2-3	FMG - Input/Output Files	10
3-1	RENUM - Input Variables and Formats	14
3-2	RENUM - Description of Input Variables and Arrays	15
3-3	RENUM - Input/Output Files	18
4-1	LINEL - Input Variables and Formats	20
4-2	LINEL - Description of Input Variables and Arrays	21
4-3	LINEL - Input/Output Files	23
5-1	ELLFMG - Input Variables and Formats	26
5-2	ELLFMG - Description of Input/Output Files	27
5-3	ELLFMG - Input/Output Files	28
A-1	FMG - Description of Program Variables	31
A-2	FMG - Description of Program Arrays	39
A-3	FMG - Subroutine Outline	42
A-4	FMG - Description of Subroutines	43
C-1	RENUM - Description of Program Variables	97
C-2	RENUM - Description of Program Arrays	99
C-3	RENUM - Subroutine Outline	101
C-4	RENUM - Description of Subroutines	102
E-1	LINEL - Description of Program Variables	123
E-2	LINEL - Description of Program Arrays	125
E-3	LINEL - Subroutine Outline	127
E-4	LINEL - Description of Subroutines	128
G-1	ELLFMG - Description of Program Variables	151
G-2	ELLFMG - Description of Program Arrays	153

Acknowledgement

The authors thank Lea Cox and Kenzi Karasaki for providing a thorough review of this report.

1.0 Introduction

The purpose of this report is to provide the user with sufficient information to run the programs FMG, RENUM, LINEL, and ELLFMG. A previous report explained the theory and the design of these programs, so that by using the two reports, a thorough understanding of the codes is possible. This report should familiarize the user with program options and modes of operation, input variables, input and output files. Information not strictly needed to run the programs, but useful in understanding their internal structure is provided in appendices. The appendices cover program variables and arrays, subroutine outlines, a short description of each subroutine, and finally listings of codes. The additional information on FMG, RENUM, LINEL, and ELLFMG is in Appendices A, C, E, G respectively, and the listings are in Appendices B, D, F, and H.

An addition program TRINET which calculates the steady state or transient flow should be used in place of LINEL. LINEL is being phased out and will not be maintained. A complete description of TRINET is in a forthcoming report.

Two simple additional programs, ELLP and DIMES, are not considered in this manual. These two programs are used to plot the networks (DIMES) output by FMG and RENUM or the permeability ellipses (ELLP) obtained by ELLFMG. These programs are machine dependent and interactive, the user specifying if he wants a plot on his screen or on a plotter.

The codes are written in FORTRAN-77 for use on a CONVEX computer. Large arrays are dimensioned by constants set in "parameter" statements. An overall maximum dimensioning parameter statement is inserted in each main program and the appropriate subroutines. For each program, the allowable problem size, controlled by the maximum number of fractures, nodes, etc., can be changed simply by editing the parameter statement and recompiling the program.

2.0 Program FMG

Several options are available for the output generated by program FMG. First, the user selects one of four modes of operation, or ways in which the program can be executed, using the input control variable **icont**:

1. Generation of primary fracture system (**icont** = 1).
2. Generation of a primary fracture system and determination of the conducting fracture system in one or more flow regions (**icont** = 2)
3. Determination of the conducting fracture system in one or more flow regions from a previously generated primary fracture system (**icont** = 4)
4. Generation of a primary fracture system, and search of a connection between sides 1 and 3 (**icont** = 5).

The shape of the generation region and the flow region are controlled by the parameter **igene**:

- Rectangular generation and flow region for directional permeability measurements
- Circular generation and flow region, with a circular hole in the flow region to model well tests.

The user then controls the computation of one or more fluid flow meshes, one for each flow region, with the input variable **imesh**.

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 2-1 indicates also which input groups are read for each value of the variable **icont**. Except for the group 2 and 3, all input variables are read from FMG.INP file. The Group 2 may also be read in the main program from the STUDY.DAT file. The group number 3 may also be read in the main program from the FRAC.DAT file, and it is unformatted; Groups 1,4,10 are read by the main program. Groups

5,6,7,8 are read by subroutine FRAGEN; Group 9 is read in subroutine SPATIA and Group 11 is read in subroutine WRENUM. Data Group 7 is repeated for each flow region; any number of flow regions can be designated for a given primary fracture system. The flow regions may be of various sizes as long as they fit within the generation rectangle, and of various orientations. Input variables are described in Table 2-1.

Table 2-3 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.

Further information about the program is given in Appendix A: description of the program variables and arrays (Table A-1 and Table A-2); a subroutine outline (Table A-3) and description of the program subroutines (Table A-4). Appendix B is a listing of the code.

Table 2-1. FMG - Input Variables and Formats

Group	Variable	Format	Input.Unit
1	icont,iplot,imesh,ipmt ikeep,iunits,igene iranf,dseed	10x,i5,3(15x,i5) " 10x,i5,15x,d15.0	1
2	istud xstud(i),ystud(i), i = 1,istud	i2 2(f6.2)	20
3* icont = 3	oray,odate,title nsets,nfrac,xgene,ygene (iseti(i,j),j=1,8), i=1,nsets (rseti(i,j),j=1,10),i=1,nsets frac(i,j),j=1,10)kut(i),i=1,nfrac	unformatted	2
4	title xgene,ygene,nsngene(igene=0) rgene,nsngene(igene=1) nsets,itole	a 2(10x,f10.4),10x,i5 10x,f10.4,30x,i5 2(10x,i5)	1
5	icent,idens	3(10x,i5)	1
6 icent = 1	nfrac frac(m,j),j=1,5;m=1,nfrac	10x,i5,15x,f10.4 5(f10.4)	1
7	nfrac (idens = 1) or rlamb, (idens = 0) ichar const(ichar = 2) or idist,ev,sd(ichar = 3) or ycept,slope,sd(ichar = 4)**	10x,i5,15x,f10.4 10x,e10.3,10x,f10.4 10x,i5 10x,f10.4 10x,i5,15x,f10.4,10x,f10.4 3(10x,f10.4)	1
8 icent = 3	rlamb ichar idist,ev,sd(ichar = 3, 6, or 7) or ycept,slope,sd(ichar = 4 or 5)**	10x,e10.3,10x,f10.4 10x, i5 10x,i5,15x,f10.4,10x,f10.4 3(10x,f10.0)	1

repeat for each
characteristic:
orientation
length, and
aperture

9 icent = 3	rlambmn(idens = 0)	10x,e10.3	7
	or		
	rlbar,rlambmn(idens = 2)	10x,e10.3,10x,f10.4	
	or		
	nfracmn(idens = 1)	10x,i5,15x,f10.4	
	const(ichar = 2)	10x,f10.4	
	or		
10 (icont=2 or 3)	ev,sd(ichar = 3)	2(10x,f10.0)	
	or		
	ycept,slope,sd(ichar = 4 or 5**)	3(10x,f10.0)	
	or		
	sd(ichar = 7)	10x,f10.0	
11	xmesh,ymesh,rotan(igene = 0)	3(10x,f10.4)	1
	rmesh,rhole,xhole,yhole(igene = 1)	4(10x,f10.4)	
11	bcode(i),i = 1,4	4i10	
	bvalu(i),i = 1,4	4f10.4	

*This group, which can be written by a previous generation, is read from a unformatted file FRAC.DAT.

**The variables ycept, slope, sd(ichar = 4) are read only when generating the apertures.

Table 2-2. FMG - Description of Input Variables

Variable	Description.
bcode(i), i=1,4	Boundary codes for side i of the flow region = -1 - constant flux = 0 - internal node = 1 - constant head = 2 - constant linearly distributed head
bvalu(i), i=1,4	Values of the boundary head or flux on side i of the flow region
const	Constant orientation, length, or aperture; const is read when ichar=2
dseed	Seed for random number generator (double-precision)
ev	Expected value (i.e., mean) of a statistical distribution
icent	Code for determining fracture characteristics; A negative value indicates that transmissivities are to be input instead of apertures. = 1 - read orientation, length, aperture and coordinates of fracture center = 2 - statistically generate coordinates of fracture center; set orientation, length, and aperture to a constant value or else statistically generate these characteristics = 3 - statistically generate fractures by zones
ichar	Code for determining individual fracture characteristics; ichar is read for each of orientation, length, and aperture/transmissivity = 2 - read constant values = 3 - generate statistically independent values = 4 - correlate aperture with log length (for aperture only) = 5 - correlate aperture with length (for aperture only)
icont	Code for controlling program execution = 1 - generate fracture system only = 2 - generate fracture system and compute the mesh = 3 - option no longer valid = 4 - read primary fracture system from local file called "frac" and compute mesh
idens	Code for reading either the density per set (rlamb) or the number of fractures per set (nfrac) = 0 - input rlamb, compute nfrac = 1 - input nfrac, compute rlamb = 2 - calculate rlamb from ribar & rlambdal (used when generating fractures by zones)

idist	Code for statistical distributions = 1 - normal distribution = 2 - lognormal distribution = 3 - exponential distribution = 4 - gamma distribution (disabled) = 5 - uniform distribution
igene	Code for the shape of the generation and flow regions = 0 - rectangular regions = 1 - circular regions, with a circular hole in the flow region.
ikeep	Code for discarding dead-ends = 0 - keep only conducting fracture network = 1 - keep dead-ends = 2 - keep dead-ends and isolated clusters
imesh	Code for producing a finite element mesh = 0 - no mesh = 1 - generate mesh
iplot	Code for producing input files for the plotting program DIMES or COPLOT = 0 - no plot = 1 - generate plot files after RENUM only = 2 - generate plot files after both FMG and RENUM
ipmt	Code for printing output in LINEL = 0 - normal output = 1 - (used only in CHANGE) = 2 - print detailed output
iranf	Code for duplicating the random number generation of a previous run = 0 - "random" choice of dseed = 1 - read dseed to duplicate previous run
istud	Number of study regions
itole	Number of decimal places in the coordinates of fracture centers
iunits	Type of units = 0 - cgs = 1 - mks
nfrac	Number of fractures per set
nfracmn	Number of fractures in a given subregion
nsets	Number of fracture sets
nsgene	Number of generation sub-regions in each direction. The total number of subregions is $(ns_{gene})^2$
rgene	radius of the generation region (igene = 1)
rhole	radius of the circular hole in the flow region (igene = 1) for well-test modeling

rlamb	Number of fractures per unit area (fracture density)
rlambmn	Density of fractures (i.e. number of fracture per unit area) in a subregion
rlbar	Mean fracture length in a subregion
rmesh	radius of the flow region (igene = 1)
rotan	Angle of rotation of the flow region
sd	Standard deviation of a statistical distribution
slope	Slope of correlation line between variables
title(2)	Title of the problem to be solved
xgene	Dimension of the generation region in x direction
xhole	x-coordinate of the center of the hole in the flow region
xmesh	Dimension of the flow region in x direction
xstud(i)	Dimension of the study region in x direction
ycept	The y intercept of correlation line between two variables
ygene	Dimension of the generation region in y direction
yhole	y-coordinate of the center of the hole in the flow region
ymesh	Dimension of the flow region in y direction
ystud(i)	Dimension of the study region in y direction

Table 2-3. FMG - Input/Output Files

Unit	Name	Read/ Write	Main Prog./ Subroutine	Description
1	FMG.INP	Read	FMG	Input data groups: 1,2,3,4,10
			FRAGEN	Input data groups: 5,6,7,8
			SPATIA	Input data group: 9
			WRENUM	Input data group: 11 Input data for plotting program:
3	LINESGR.DAT	Write	PLOCOO	Fractures in generation region (if iplot = 1 or 2)
3	LINES00.DAT	Write	PLOCOO	Fractures in first flow region before discarding anything (if iplot = 1 or 2)
3	LINESnn.DAT	Write	PLOCOO	Fractures in flow region for each rotation (if iplot = 2)
4	RENUMGR.DAT	Write	PLOCOO	Header for generation region
4	RENUM00.DAT	Write	PLOCOO	Header for first flow region
4	RENUMnn.DAT	Write	WRENUM	Input data for RENUM pro- gram for each flow region
5	FRAC.TXT	Write	WRENUM	Arrays frac(mfrac,10) and kut(mfrc)
6	FMG.OUT	Write	PFS	Statistics on characteristics of fractures by sets
			WRENUM	Titles and data about flow region
7	SUBREG.DAT	Read	SPATIA	Input data group: 9

FRAGEN

8	FRAC.DAT	Write	FMG	All primary fracture system data: file is written when icon = 1 or 2
8	FRAC.DAT	Read	FMG	All primary fracture system data: file is read in for each flow region in the run when icon = 1 or 2 or in subsequent runs when icon = 4, input data group 3
20	STUDY.DAT	Read	FMG	Input data group: 2
50	CONNECTIONS	Read/Write	CONNEC	Number of runs for which a connection has been found (icon = 5)

3.0 Program RENUM

The user does not need to write any input to RENUM. All the information the program needs is contained in the files RENUMnn.DAT created by FMG or CHANGE. Subroutine RDATA reads input from RENUM01.DAT, RENUM02.DAT, RENUM03.DAT, etc. MERGE then discards zero length elements, by combining nodes very close together into one node. Paths connected to the boundaries are traced in subroutine PATHS, in order to discard any complex dead-ends. Then the nodes are renumbered using a modified Cuthill-McKee method preparing them for output. This is done in subroutine MCGEE. A plot file is written by subroutine PLOT and the input file for the finite element program LINEL, LINEL.INP, is then output.

Alternatively, if the input file INTER.INP exists in the directory the input files for TRINET are written: CTRL.INP, ELMT.INP, NODE.INP. See TRINET manual for description of program.

Table 3-1 lists the input variables and their formats. These variables are described in Table 3-2. Table 3-3 lists the Input/Output units and files used by the program, describes succinctly the content of each file, and identifies the subroutines that read or write data. Further information about the program is given in Appendix C: description of the program variables and arrays (Table C-1 and Table C-2); a subroutine outline (Table C-3) and a description of each subroutine (Table C-4). Appendix D is a listing of the code.

Table 3-1. RENUM - Input Variables and Formatting

Variable	Format	Input Unit
title	10(a/)	1
neleme,nnodes,maxd,	3(10x,i6,5x)	
ikeep,iplot	2(10x,i5,5x)	
(ibs(1,m),ibs(2,m),ifrac(m),(xyzphi(i,m),i=1,6),m=1,nnodes)	5x,3i5,6f10.0	
(inode(1,n),inode(2,n),aptdis(1,n),aptdis(2,n),ifr(1,n), ifr(2,n),n=1,neleme)	5x,2i5,5x,2e10.4,5x,2i5	
If INTER.INP exists: (control variable for TRINET)		3
title2	a80	
imode,bmod,mxstep	3(10x,i5)	
time0,tmax,dtini	3(10x,f10.0)	
prr,theta,tolc	3(10x,f10.0)	
dcint,dcon,dcoff	3(10x,f10.0)	
rhorr,mur,gravr	3(10x,f10.0)	
ssubs,dispc	2(10x,f10.0)	

Table 3-2. RENUM - Description of Input Variables and Arrays

Variable/Array	Description
aptdis(2,mxelem)	Fracture characteristics: aptdis(1,m)= transmissivity of element m aptdis(2,m)= length of element m
dcint	If the concentration difference is less than dcint between two adjacent nodes found during upstream search in btrack.f, cubic spline interpolation is used in cubeint.f to obtain the concentration of the originating fixed node, otherwise upwind scheme is used.
dcoff	When the concentration difference between two nodes becomes less than dcoff, and if one of them is a moving node, it is deleted in pluck.f.
dcon	When the concentration difference between the node and an adjacent node is larger than dcon, a moving node(s) is originated from the node in ftrack.f.
dtini	Initial time step increment, Δt .
gravr	Gravitational constant. Default is 9.8067 m/sec ² .
ibs(2,mxnode)	Node information; n = 1, nnodes ibs(1,n) = node type = 0 internal node = 1 imposed head boundary node = -1 imposed flux boundary node ibs(2,n) = boundary side number for boundary nodes
ibmode	Used in renum.f to determine the boundary from which Cuthil-McKee renumbering scheme starts: ibmode = 0 Radial boundary (start from side 5 only), 1 Square boundary (start from side 2 only), 2 Square boundary (start from all four sides: 1, 2, 3, 4).
ifrac(mxnode)	Number of fractures on which the node exists (3-D only).
ifr(2,mxelem)	Fracture numbers ic = 1, neleme

	<p>ifr(1,ie) = number of the fracture on which lies an element (fracture line in 2-D; fracture disc in 3-D)</p> <p>ifr(2,ie) = number of other fracture disc on which lies an element, if relevant (3-D only)</p>
ikeep	<p>Code for discarding dead-ends</p> <p>= 0 - keep only conducting fracture network</p> <p>= 1 - keep dead-ends</p> <p>= 2 - keep dead-ends and isolated clusters</p>
imode	<p>Used to control types of problems:</p> <p>imode = -2 Steady-state flow only (theta = 1.0),</p> <p>-1 Transient flow only,</p> <p>0 Test data deck,</p> <p>1 Transport in transient flow field,</p> <p>2 Transport in steady-state flow field (theta = 1.0).</p>
iplot	<p>Code for producing input files for (fracture) the plotting program DIMES</p> <p>= 0 - no plot</p> <p>= 1 - generate plot files after RENUM only</p> <p>= 2 - generate plot files after both FMG and RENUM</p>
inode(2,mxclem)	<p>Node number of the two nodes defining an element</p> <p>m = 1, neleme</p> <p>inode(1,m) = node number of the first node of element m</p> <p>inode(2,m) = node number of the second node of element m</p>
maxd	Number of sets of boundary conditions
mxstep	Maximum time step desired in the simulation.
neleme	Number of elements
nnodes	Number of nodes
prr	<p>By default, the time step Δt is increased geometrically using the following formula:</p> $\Delta t_{n+1} = \Delta t_n \times prr$
rho	Density of fluid. Default value is 1000 m ³ /kg.
mmur	Dynamic viscosity of fluid. Default value is 0.001 kg/m-sec.

theta	<p>Time weighting constant used in the following manner. Recommended value is 0.66. For steady-state flow problem, it is set to 1 internally.</p> $\left[\frac{[A]^t}{\Delta t} + [B]^t \right] \cdot \{h\}^{t+\Delta t} = \{F\} + \left[\frac{[A]^t}{\Delta t} - (1 - \Theta \cdot [B])^t \right] \cdot \{h\}^t,$ <p>where [A] is the storage coefficient matrix, [B] is the permeability matrix, {F} contains boundary conditions, {h} is the head vector. theta is used analogously for mass diffusion equation.</p>
time0	Initial time of the start of simulation. time0 is other than zero when restarting a simulation which is stopped in the middle.
title	Title of the simulation run. Three lines are used. Two title lines are inherited from the mesh generator. One line is added from INTER.
title(10)	First ten lines read from RENUM%.DAT then written to LINEL.INP; information needed by LINEL but not by RENUM. (character*80)
tmax	Maximum simulation time allowed (in real time).
tole	Tolerance to allow for numerical round off. For example, if the distance between two nodes are less than tole they are merged to one node.
xyzphi(6,mxnode)	<p>Node information:</p> <p>n1 = 1, nnodes</p> <p>xyzphi(1,n)= x-coordinate of node n</p> <p>xyzphi(2,n)= y-coordinate of node n</p> <p>xyzphi(3,n)= z-coordinate of node n</p> <p>xyzphi(4,n)= value of imposed head at node n, first boundary conditions</p> <p>xyzphi(5,n)= value of imposed head at node n, second boundary conditions</p> <p>xyzphi(6,n)= value of imposed head at node n, third boundary conditions</p>

Table 3-3. RENUM - Input/Output Files

Unit	Name	Read/ Write	Main. Prog/ Subroutine	Description
1	RENUM%%.DAT	Read	RDATA	Input to program RENUM written by program FMG (see Table 1-1)
3	INTER.INP	Read	RCNTRL	Control variables for input to program TRINET
4	LINEL.INP	Write	PROUT	Node and element information; input to program LINEL
7	LINES%%.DAT	Write	PLOT	Plotting files used by program DIMES to make fracture plots of the flow regions
8	CTRL.INP	Write	TRIOUT	Control variables; input to program TRINET
9	NODE.INP	Write	TRIOUT	Node information; input to program TRINET
10	ELMT.INP	Write	TRIOUT	Element information; input to program TRINET
11	RENUM.ERR	Write	WERROR	Run-time error messages
12	NPN.INP	Read	RNPN	Read only if NPN.INP exists (see TRINET manual)

4.0 Program LINEL

In the same way as RENUM, the user does not need to write any input to LINEL. All the information the program needs is contained in the files LINEL.INP and STUDY.DAT. First, the study regions data is read by RSTUD. The header of the data file is read by RHEAD, and the nodes and elements information is read by RMESH. SORTU sorts the two nodes that form each element and then sorts the elements by their first node number. CPHI fills the matrix and vector constituting the linear system of equations to be solved. Then SYMSOL solves the system using a banded lower triangle decomposition. From the head at each node, the flux entering or leaving the network at each imposed head boundary node is computed by CUNK. PINFO print the header of the output file LINEL.OUT. SFLUX sums up and prints the fluxes on the various boundaries. The head at imposed flux boundary nodes is also output. If study regions were specified, the heads and fluxes at their boundaries are computed and printed by subroutine STUDY. This program is being phased out in favor of the transient flow program TRINET. See TRINET manual for details.

Table 4-1 lists the input variables and their formats. These variables are described in Table 4-2. Table 4-3 lists the Input/Output units and files used by the program, describes succinctly the content of each file, and identifies the subroutines that read or write data. Further information about the program is given in Appendix E: Description of the Program Variables and Arrays (Tables E-1 and E-2); a subroutine outline (Table E-3) and a description of each subroutine (Table E-4). Appendix F is a listing of the code.

Table 4-1. LINEL - Input Variables and Formats

Group	Variable	Format	Input.Unit
1	istud,igrad (xstud(k),ystud(k),k=1,istud)	2i2 2f6.2	20
2	jobnam,date,ipmt title nfrac xgene,ygene,zgene xmesh,ymesh,zmesh rotan,rotan2 ibcode bvalu visc,spgr neleme,nnodes,maxd ibwth	a19,3x,a9,i1 a/a 10x,i5 3(10x,f10.4) 3(10x,f10.4) 2(10x,f10.4) 10x,6i10 10x,6f10.0 2(10x,f10.0) 3(10x,i6,4x) 50x,i5	1
3	(ib(m),side(m),(coord(i,m),i=1,3) (phi(m,irot),irot=1,maxd),m=1,nnodes)	5x,2i5,5x,6f10.4	1
	(inode(1,n),inode(2,n),trans(n), dist(n),n=1,neleme)	5x,2i5,5x,2e10.4	1

Table 4-2. LINEL - Description of Input Variables and Arrays

Variable	Description
bvalu(i),i=1,6	Values of the boundary head or flux on side i of the flow region (bvalu(5) = bvalu(6) = 0 for 2-D cases)
coord(3,mxnode)	node coordinates, n=1, nnodes coord(1,n) = x coordinate of node n coord(2,n) = y coordinate of node n coord(3,n) = z coordinate of node n (0 if 2-D)
dist (mxelem)	Element length array ie=1, neleme dist(ie) = length of element number ie
ib(mxnode)	node type, n=1, nnodes ib(n) = 0 internal node = 1 imposed head boundary node = -1 imposed flux boundary node
ibcode(i),i=1,6	Boundary codes for side i of the flow region (ibcode(5) = ibcode(6) = 0 for 2-D cases) = -1 - constant flux = 0 - internal node = 1 - constant head = 2 - constant linearly distributed head
ibwth	Bandwidth of the linear system
igrad	Type of gradient to be used for study region permeability calculations = 0 use both types of gradients = 1 use global gradient only = 2 use local gradient only
inode(2,mxelem)	Element-node reference array, ie = 1, neleme inode(1,ie) = number of the node making up the first endpoint of element number ie inode(2,ie) = number of the node making up the second endpoint of element number ie
ipmt	Code for printing output = 0 print normal output in LINEL.OUT = 1 print normal output plus heads at disc intersections in LINEL.OUT (3D only) = 2 print detailed output in LINELALL.OUT
istud	Number of study regions

jdate	Character variable containing the date at which the line network was generated
jobnam	Character variable identifying the program that generated the line network
maxd	Number of different boundary conditions to be solved for the same network
neleme	Number of elements in the network
nfrac	Number of fractures in the flow region
nnodes	Number of nodes in the network
phi(mxnode,3)	Imposed head or flux on a node, if relevant, $n = 1, nnodes$; $irod = 1, maxd$ $phi(n,irod) = 0$ if node n is internal $phi(n,irod) =$ value of imposed head or flux for boundary node number n for set number $irod$ of boundary conditions
rotan,rotan2	Rotation angles of the flow region($rotan2 = 0$ for 2-D cases) For circular flow region, $rotan$ and $rotan2$ are the coordinates of the center of the hole
side(mxnode)	Boundary side number, $n=1, nnodes$ $side(n) =$ side on which boundary node is lying $= 0$ if internal node
spgr	Specific gravity in the unit system chosen by the user
title(2)	Character*80, title of the problem to be solved
trans(mxelem)	Element transmissivities array $ie = 1, neleme$ $trans(ie) =$ transmissivity of element number ie
visc	Dynamic viscosity of water in the unit system chosen by the user
xgene,ygene,zgene	Size of the generation region ($zmesh = 0$ for 2-D cases) for circular generation regions, $ygene = 0$, and the radius is read in $xgene$
xmesh,ymesh,zmesh	Size of the flow region ($z mesh = 0$ for 2-D cases) for circular flow regions, $xmesh =$ radius of flow region $ymesh =$ radius of hole
xstud(20)	Size of the study regions in the x direction
ystud(20)	Size of the study regions in the y direction

Table 4-3. LINEL - Input/Output Files

Unit	Name	Read/ Write	Main Prog./ Subroutine	Description
1	LINEL.INP	READ	RHEAD RMESH	Input data Group: 3,4,5
6	LINEL.OUT	WRITE	CPHI,PINFO	-
8	LINELALL.OUT	WRITE	PINFO RMESH	-
10	LINEL.ERR	WRITE	WERROR	-
11	ELLIPSE.INP	WRITE	SFLUX	-
20	STUDY.DAT	READ	RSTUD	Input data group: 1,2
31	ELLIPSEG01.INP	WRITE	STUDY	
32	ELLIPSEG02.INP	"	"	
.	"	"	"	
.	"	"	"	
.	"	"	"	
[istud+30]	ELLIPSEG[istud].INP	"	"	
51	ELLIPSEI01.INP	WRITE	STUDY	
52	ELLIPSEI02.INP	"	"	
.	"	"	"	
.	"	"	"	
.	"	"	"	
[istud+50]	ELLIPSEI[istud].INP	"	"	

5.0 Program ELLFMG

The user does not need to write any input to ELLFMG. All the information the program needs is contained in the files ELLIPSE.INP, ELLIPSEG01.INP, ELLIPSEL01.INP, etc. These files are written by program LNEL. ELLFMG computes the best fit ellipse for a given set of directional permeability measurements. Table 5-1 lists the input variables and their formats. These variables are described in Table 5-2. Table 5-3 lists the input and output files used by the program. A description of the program variables and arrays is given in Appendix G (Table G1 and G-2). Appendix H is a listing of the code.

Table 5-1. ELLFM_{1.1} - Input Variables and Formats

Group	Variable	Format	Input Unit
1	iray, idate jobname, jdate title(4)	a7,5x,a9 a7,5x,a9 a80	11
2	angle, xkganre, pkganre (as many lines as there are permeability measurements)	3e15,5	11

Table 5-2. ELLFMG - Description of Input Variables

Variable	Description
angle	Angle between x-axis and the direction of the gradient for a permeability measurement
idate	Date of creation of the fracture network
iray	Name of the program which created the network
jdate	Date the flow was computed
jobname	Name of the job which computed the flow
pkganre	Inverse of the square root of permeability for gradient direction "angle"
title(4)	Title lines for identifying the run
xkganre	Permeability for gradient direction "angle"

Table 5-3. ELLFMG - Input/Output Files

Unit	Name	Read/ Write	Main Program Subroutine	Description
6	ELLFMG.OUT	WRITE	Main	Output file for ELLFMG.
7	ELLIPSE.PLT	WRITE	Main	File containing plotting information
7	ELLIPSEG01.PLT ELLIPSEG02.PLT 03 PLT 04 : (istud)	WRITE WRITE	Main Main	Files containing plotting information for global gradients study regions if relevant*
7	ELLIPSEL01.PLT 02 03 : (istud)	WRITE	Main	Files containing plotting information for local gradient study regions if relevant*
11	ELLIPSE.INP (if ngrad = 0)	READ	Main	Input file for ELLFMG (flow region)
11	ELLIPSEG01.INP 02 03 : (istud) if(ngrad=1)	READ	Main	Input file for ELLFMG (global gradient study regions if relevant*)
11	ELLIPSEL01.INP 02 03 : (istud) if (ngrad = 2)	READ	Main	Input file for ELLFMG(local gradient study region if relevant*)

* see Theory and Design report, Chapter 4 (LINEL), for definition of local and global gradient study regions

Appendix A

FMG - Program Organization and Arrays

Table A-1. FMG - Description of Program Variables

Variable	Description	How Value is Assigned
Global:		
idate	Current date, character*9	Set by calling FORTRAN subroutine DATE
igene	Code for the shape of the generation and flow regions = 0 rectangular regions = 1 circular regions, with a circular hole in the flow region	Read from FMG.INP in main program
ikeep	Code for discarding dead-ends = 0 - keep only conducting fracture network = 1 - keep dead-ends = 2 - keep dead-ends and isolated clusters	Read from FMG.INP in main program
imesh	Code for producing a finite element mesh = 0 - no mesh = 1 - generate mesh	Read from FMG.INP in main program
ipmt	Parameter for LINEL passed to RENUM %%.DAT	Read from FMG.INP in main program
iranf	Code for duplicating the random number generation of a previous run = 0 - "random" choice of dseed = 1 - read dseed to duplicate previous run	Read from FMG.INP in main program
iray	Label for output file (FMG.OUT), character*19	Set in main program
istud	The number of study regions	Main program, PLOCOO
itole	Number of decimal places in the coordinates of fracture centers	Read from FMG.INP in main program
iunits	Type of units	Read from FMG.INP in

	= 0 - cgs = 1 - mks	main program
lplot	Keeps track of the number of the file where to write the plot information	Subroutine WRENUM
lrenum	Keeps track of the number of the file where to write mesh data for RENUM	Subroutines WRENUM, PLO-COO
mfrc	Maximum number of fractures	Set in a parameter statement in main program
mnod	Maximum number of nodes	Set in a parameter statement in main program
mkey	mnod*2	Set in a parameter statement in main program
nfrac	Number of fractures per set, then total number of fractures	Read from FMG.INP if idens=1; calculated in FRAGEN if idens=0 reset in limit, connec
nsets	Number of fracture sets	Read from FMG.INP in main program
nsgene	Number of sub-generation regions in each direction. The total number of subregions is (nsgene) ²	Read from FMG.INP in main program
pi180	Conversion factor from degrees to radians	Set in main program
qc	Factor which multiplied by the cubic power of the aperture gives the transmissivity	Set in main program
rmesh	Radius of the flow region. Set to rgene for the first call to subroutine CLIMIT in order to truncate fractures at the generation region boundaries	Read from FMG.INP in main program
rgene	Radius of the generation region (igene = 1)	Read from FMG.INP in main program
rhole	Radius of the circular hole in the flow region (igene = 1, for well-test modelling)	Read from FMG.INP in main program

rotan	Angle of rotation of the flow region	Read from FMG.INP in main program
spgr	Specific gravity	Set in main program according with the type of units used
visc	Viscosity	Set in main program
xgene	Width of the generation region in x-direction	Read from FMG.INP in main program
xhole	x-coordinate of the center of the hole in the flow region (igene = 1)	Read from FMG.INP in main program
xmesh	Width of the flow region in x-direction Set to xgene for the first call to subroutine RLIMIT in order to truncate fractures at the generation region boundaries	Read from FMG.INP in main program
ygene	Width of generation region in y-direction	Read from FMG.INP in main program
yhole	y-coordinate of the center of the hole in the flow region (igene = 1)	Read from FMG.INP in main program
ymesh	Width of flow mesh region in y-direction Set to ygene for the first call to subroutine RLIMIT in order to truncate fractures at the generation region boundaries	Read from FMG.INP in main program
Local:		
ai	Variable assigned to frac(i,8). See arrays description	Defined in CONNEC
ainf	Minimum of the range of a uniform distribution	Computed in UNIFOD
aj	Variable assigned to frac(j,8) See arrays description	Defined in CONNEC
alen	Length of the fracture	CLIMIT
bi	Variable assigned to frac(i,9) - see array description	Defined in subroutine CONNEC
bj	Variable assigned to frac(j,9) - see array description	Defined in subroutine CONNEC

delt	Discriminant of second order equation	CLIMIT
file	Char*7 File name	Defined in main program, subroutine SPATIA and WRENUM
filel	Char*7 File name	Used in subroutine PLOCOO
filer	Char*7 File name	Used in subroutine PLOCOO
ci	Variable assigned to $\text{frac}(i,10)$. See array description.	Defined in subroutine CONNEC
cj	Variable assigned to $\text{frac}(j,10)$.	Defined in subroutine CONNEC
cosr	Variable assigned to the cosine of the rotation angle of the flow region	Defined in WRENUM
cov	Covariance between two variables	Dummy argument in subroutine FRSTA1
crc	Correlation coefficient	Appear in subroutine FRSTA1
dist	The length of the element	Used in subroutine WRENUM
dx	One half of the projection of the length of fracture i on the x axis	Defined in subroutine ENDPTS
dy	One half of the projection of the length of fracture i on the y axis, taken with minus sign	Defined in subroutine ENDPTS
eapt	Dummy argument in subroutine FRSTA1 which takes the expected value of the statistical distribution for $k = 1$	
elen	Dummy argument in sub FRSTA1 which has the value of the expected statistical distribution for $k = 2$	
emin	Minimum value of the parameter	Set and used in NORMD1
flen	Length of fracture i	Subroutine LIMIT
hlen	Half of the length of fracture i	Defined in subroutine ENDPTS
ialpha	Integer form of $[\alpha]$	Subroutine ORIENT
ibkeyl	Value of boundary codes for sides $i=1,4$	Subroutine WRENUM

ibkey2	Value of boundary code for sides i = 1,4	Subroutine WRENUM
ieleme	Element number	Subroutine WRENUM
ier	Error code on return from the subroutine RLLAV	Subroutine FRSTA1
ifrc	Other fracture of intersection	Subroutine CONNEC
ifrc1	The number assigned to the first fracture in the current level	Subroutine CONNEC
ifrc2	Last fracture in current level	Subroutine CONNEC
ifrc3	Last fracture in next level	Subroutine CONNEC
ilevel	Current level	Subroutine CONNEC
in1	Node number 1 of current element	Subroutine WRENUM
in2	Node number 2 of the current element	Subroutine WRENUM
ind	Number of the fracture to be discarded	Subroutine CONNEC
indint	Index into the k node array	Subroutine CONNEC
inext	Pointer used to build the array next	Subroutine LIMIT
inode	Node number counter	Subroutine WRENUM
int1	First index in the [knode] array	Subroutine WRENUM
int2	Last index in the [knode] array	
io	Old number for fracture i	Subroutine CONNEC and WRENUM
iside	Side for the boundary codes	Defined in subroutine WRENUM
jint1	Index used in computing the intersections with previous fractures	Subroutine CONNEC
jint2	Index used in computing the intersections with previous fractures	Subroutine CONNEC

k0	Index of the first fracture in the next level	Subroutine CONNEC
k1	The new fracture number	Subroutine CONNEC
k2	Keeps track of the index of the next fracture to be considered	Subroutine CONNEC
ku	Truncation code (see array kut)	CLIMIT
maxd	Number of sets of boundary conditions output by the program (2 if constant gradient boundary conditions are specified, 1 if any imposed flux condition is specified)	Subroutine WRENUM
maxfrc	Maximum number of fractures	
msk1	Integer*2. Variable assigned to mask (1,i) while checking for intersections with fractures not previously included	Subroutine CONNEC
msk2	Integer*2. Variable assigned to mask (2,i) while checking for intersections with fractures not previously included	Subroutine CONNEC
ncon	Number of runs with connections	CONNEC
neleme	Number of elements	Subroutine CONNEC
nextra	Number of extra elements and nodes for non-truncated meshes (ikeep > 0)	Subroutine CONNEC
nslice	Dimensioning parameter for local arrays	Parameter statement in ORIENT
nf	Number of fractures in a set	Subroutine PFS
nfracmn	Number of fractures in a given subregion	Subroutine SPATIA
nnodes	Number of nodes	Subroutine CONNEC
nt	The number of intersections of a fracture line with the boundary lines of the flow region	Subroutine LIMIT
prtx	The new x coordinate of one end of the fracture	Subroutine LIMIT

prty	The new y coordinate of one end of the fracture	Subroutine LIMIT
ran	Random number	Subroutine RANDXY
rlambdal	Linear density, used to compute rlamblmn if dens = 2	Subroutine SPATIA
rlamblmn	Number of fractures in a subregion	Subroutine SPATIA
ribar	Mean fracture length in a subregion. Read from the file SUBREG.DAT.	Used in subroutine FRAGEN
rhsq	r_{hole}^2	CLIMIT
rmsq	r_{mesh}^2	CLIMIT
rs	Radius of generation region	CIRCXY
sdn	Standard deviation of normal distribution associated to log normal distribution with parameters cv,sd	Subroutine LOGNOD
sinr	Dummy variable assigned to sine of the rotation angle	Defined in subroutine WRENUM
slen	Standard deviation of the fracture length for the set under consideration	Subroutine FRSTA1
sn	Sum of 25 random variables distributed uniformly in (0,1)	Subroutine NORMAD
t1	Distance between the first end point of the fracture and a node	Subroutine WRENUM
t1,t2	Relative coordinates along the fracture line of its intersections with a circle (generation region, flow region, or hole). The relative coordinate is zero at the first endpoint and one at the second endpoint.	CLIMIT
told	[t] value of previous node used to compute element length	Subroutine WRENUM
toler	Tolerance in the minimum distance between points	Subroutine CIRCXY, RECTXY, CONNEC, WRENUM

transm	Transmissivity	Subroutine WRENUM
u0	x coordinates of the resultant vector in computing the standard deviation of orientation	Subroutine ORIEST
v0	y coordinate of the resultant vector in computing the standard deviation of orientation	Subroutine ORIEST
x1	x coordinate of first endpoint of fracture	CLIMIT
x2	x coordinate of second endpoint of fracture	CLIMIT
xc	x coordinate of fracture center	Subroutine ENDPTS
xg2	Half of the width of the generation region in the x direction	Subroutine SPATIA
xm2	Half of the width of the flow mesh region in the x direction	Subroutine LIMIT
xsubgene	Width of the generation subregion in x direction	Subroutine SPATIA
y1	y coordinate of first endpoint of fracture	CLIMIT
y2	y coordinate of second endpoint of fracture	CLIMIT
yc	y coordinate of fracture center	Subroutine ENDPTS
yg2	Half of the width of the generation region in the y direction	Subroutine SPATIA
ym2	Half of the width of the flow mesh region in the y direction	Subroutine LIMIT
ysubgene	Width of the generation subregion in y direction	Subroutine SPATIA

Table A-2. FMG - Description of Program Arrays

Array	Description
Global:	
frac(mfr,10)	<p>Fracture characteristics, read, set, or generated in subroutine FRAGEN,</p> <p>$n=1, n_{frac}$</p> <p>frac(i,1) = orientation</p> <p>frac(i,2) = length</p> <p>frac(i,3) = aperture</p> <p>frac(i,4) = x1, x coordinate of fracture center or end</p> <p>frac(i,5) = y1, y coordinate of fracture center or end</p> <p>frac(i,6) = x2, x coordinate of fracture end</p> <p>frac(i,7) = y2, y coordinate of fracture end</p> <p>frac(i,8) = a, = $-\sin(\text{orientation})$</p> <p>frac(i,9) = b, = $\cos(\text{orientation})$</p> <p>frac(i,10) = c, = $-(a*x1 + b*y1)$</p> <p>so that $ax + by + c = 0$ is the equation of the line supporting the fracture.</p> <p>Note that frac(i,4) and frac(i,5) are first the fracture center coordinates during generation. They are then set to the coordinates of the first end-point of the fracture in subroutine ENDPTS.</p>
ifrac(2,mnod)	<p>The numbers of the two fractures that determine a node. Used in subroutines CONNEC and WRENUM.</p>
iseti(20,8)	<p>Used in subroutines FRAGEN and SPATIA to store the values of nfrac, icent, and idist for each set.</p>
kut(mfr)	<p>Truncation code for each fracture. Set up in CLMIT or RLIMIT. If fracture in cuts sides i and j (where i and j are 0 when no side is cut), then $kut(in) = 16*i + j$.</p> <p>In subroutine CONNEC, [kut] is then transformed into an index for the array [knode].</p>
knode(mkey)	<p>Index into the arrays [ifrac] and [tint].</p>
mask(2,mfr)	<p>Mask resulting from bit mapping of fractures. Used to save time in computing intersections.</p>
next(mfr)	<p>Pointer keeping track of the fractures that intersect the flow region sides.</p>

rseti(20,11)	Used in subroutines FRAGEN and SPATIA to store the values of const, ev, and sd for each set.
tini(2,mnod)	Distance between the first endpoint of a fracture and the node.
xstud(10)	Dimension of the study region in x direction.
ystud(10)	Dimension of the study region in y direction.
Local:	
a(n)	Array used to store the random distribution once this is created. Used in subroutine DISTRJ and in all the random generation subroutines.
bcode(i)	Boundary codes for sides i=1,4 of the flow region which are read from FMG.INP = -1 - constant flux = 0 - internal node = 1 - constant head = 2 - constant, head linearly distributed
bval(2)	The heads for the rotation and for rotation plus 90° of the boundary node.
bvalu(4)	Boundary values of the head or flux on side i=1,4 of the flow region.
corner(2,5)	Coordinates of the corners of the flow region.
is(4)	The number attached to each flow region boundary line. Can take the values 1,2,3 and 4.
mseed(3)	Seeds for random number generator.
ola(3)	Character *11 array to store the words "orientation", "length", "aperture".
rseed(3)	Seeds for random number generator.
rseti(20,11)	Used in subroutines FRAGEN, SPATIA to store the constant value (ichar = 2); or expected value and standard deviation of each parameter for each set.
type(5)	Stores the name of the type of distribution.
wk(4000)	Temporary working storage for IMSL.

x(n)	X coordinate of randomly distributed fracture centers in RECTXY and CIRCXY.
xy(mfrc,6)	Contains two variables (length and aperture) and three work spaces for RLLAV.
y(n)	Y coordinates of randomly distributed fracture centers in RECTXY and CIRCXY.

Table A-3. FMG - Subroutine Outline

FMG	FRAGEN	SPATIA	RANDXY NORMD1 DISTR1	NORMAD LOGNOD EXPOND UNIFOD
		RECTXY CIRCXY NORMD1 DISTR1	NORMAD LOGNOD EXPOND UNIFOD	
	EQLINE ENDPTS PFS	ORIENT FRSTAT FRSTA1	RLLAV	
	RLIMIT CLIMIT PLOCOD CONNEC WRENUM	MOVE		

Table A-4. FMG - Description of Subroutines

CIRCXY	<p>Generates the coordinates of the fracture centers according to a random distribution in a circle.</p> <p>given: n,dseed,rs,itole</p> <p>returns: x(n),y(n)</p>
CLIMIT	<p>Truncates the portions of the fractures lying</p> <ul style="list-style-type: none">- outside of the generation region at the first call; or- outside of the flow region or inside of the hole at the second call. <p>Recalculates the fracture endpoints and length. Fills the [mask(2,mfrc)] and [kut(mfrc)] arrays.</p> <p>given: nfrac,nsets,iseti(nsets,2),maxfrc, (frac(nfrac,i), i=4 to 7),rmesh</p> <p>returns: frac(nfrac,2),(frac(nfrac,i),i=4 to 7), kut(nfrac),mask(2,nfrac)</p> <p>uses: MOVE</p>
CONNEC	<p>Searches for intersections between fractures. Starts with the initial fractures selected by subroutine RLIMIT or CLIMIT. Looks for fractures intersecting the initial ones, then fractures intersecting the ones it found, and so on. If icon is not 5, stops when there is no more intersection to be found (ikeep = 0) or when all fractures have been checked (ikeep ≥ 1). If icon is 5, stops whenever boundary side 3 is reached, or when there is no more intersection to be found. Calculates the number of nodes and elements.</p> <p>given: maxfrc,frac(mfrc,10),kut(mfrc),nfrac,itole,ikeep mask(2,mfrc),ifrac(2,mnod),next(mfrc)</p> <p>returns: tint(2,mnod), modified kut, next</p> <p>prints: ncon</p>
DISTRI	<p>Calls the appropriate distribution routine to be used in the generation of values, based upon the value of the code [idist] such that if:</p> <p>idist = 1 - normal distribution = 2 - lognormal distribution = 3 - exponential distribution</p>

= 4 - gamma distribution (disabled)
= 5 - uniform distribution

given: idist

returns: a(n)

uses subroutines: NORMAD
 LOGNOD
 EXPOND
 UNIFOD

ENDPTS Takes the fracture characteristics (orientation, length and center coordinates) and computes the coordinates of each endpoint of the fracture.

given: maxfrc,nfrac,frac(i,1),frac(i,2),frac(i,4),frac(i,5)

returns: frac(i,4),frac(i,5),frac(i,6),frac(i,7)

Note that frac(i,4) and frac(i,5) are modified in ENDPTS.

EQLINE Calculates the coefficients a, b and c of the line which supports each fracture.

given: maxfrc,nfrac,frac(i,1),frac(i,4),frac(i,5)

returns: frac(i,8),frac(i,9),frac(i,10)

EXPOND Generates random variables distributed exponentially with expected value ev.

given: n,dseed,ev

returns: a(n)

FRAGEN Reads in or generates the following fracture characteristics: orientation, length, aperture, and the coordinates of the fracture center.

given: maxfrc,dseed,nsets,xgene,ygene,nsgene,iranf,
 icent,idents,ipois,ltol,iseti(i,j),rseti(i,j),rgene,igene

reads: icent,idents,ipois,nfrac,theta
 [frac(m,j); j=1,5; m = n1,n2],rlamb,ichar,const,
 idist,ev,sd,ycept,slope,sd

returns: nfrac, (frac(i,j), j = 1,5), i = 1, nfrac

uses subroutines:

SPATIA	RECTXY	
	NORMD1	
	DISTRI	NORMAD
		LOGNOD
		EXPOND
		UNIFOD
RECTXY		
CIRCXY		
NORMD1		
DISTRI	NORMAD	
	LOGNOD	
	EXPOND	
	UNIFOD	

FRSTAT Calculates a mean [ev] and a standard deviation [sd]

given: a(n),n,k

returns: ev,sd

FRSTA1 Calculates the covariance [cov] and correlation coefficient [crc] if the index ichar = 4 or ichar = 5 (which corresponds with having aperture and length correlated when the fracture characteristics are generated).

given: n,a(n),b(n),ichar,elen,slen,eapt,sapt,beta(1), beta(2)

returns: cvc,crc,ycept,slope

GGUBFS: random number generator

given: dseed

returns: ggubfs, modified dseed

LOGNOD Generates random variables distributed lognormally with mean [ev] and standard deviation [sd].

given: ev,sd,dseed,n

returns: a(n)

MOVE Moves the information in the array frac for all fractures with number greater than i. This subroutine is used to make room for boundary fractures newly created because of the splitting of fractures at an inner boundary (hole).

	<p>given: i,nfrac,n2,frac(nfrac,10)</p> <p>returns: modified i,nfrac,n2,frac</p>
NORMAD	<p>Generates random variables distributed normally with expected value [ev] and standard deviation [sd].</p> <p>given: ev,sd,dseed,n</p> <p>returns: a(n)</p>
NORMD1	<p>Generates random variables distributed normally with expected value [ev] and standard deviation [sd], where [ev] for one parameter (i.e. aperture) is proportional to the logarithm of another parameter (i.e. length) or to the parameter itself depending upon the value of the parameter [ichar] (i.e., if ichar = 4 use log (length), if ichar = 5 use length).</p> <p>given: ev,sd,dseed,ichar,ycept,slope,b(n)</p> <p>returns: a(n)</p>
ORIENT	<p>Calculates the basic statistic elements (mean and circular standard deviation) for orientation distributions.</p> <p>given: a(n),b(n),d(n),nslice</p> <p>returns: ev,sd</p>
PFS	<p>Calculates and prints the fracture statistics for each set in order to compare them to the specified fracture statistics.</p> <p>given: frac(maxfrc,10),nsets,iseti(20,8),rseti(20,11)</p> <p>prints: ev,sd</p> <p>uses subroutines: ORIENT FRSTAT FRSTA1</p>
PLOCOO	<p>Writes the input decks for the plotting programs DIMES or COPLLOT</p> <p>given: maxfrc,frac(maxfrc,10),itole,iranf, iunits,ikeep,imesh,iskip8,nfrac, iplot,lplot,xgene,ygene,xmesh,ymesh, rotan,nsgene,istud,xstud(10),ystud(10)</p>

prints: fracture endpoints

RECTXY Generates the coordinates of the fracture centers according to a random distribution in a rectangle.

given: n,dseed,xs,ys,itole,ipois,slcos,slsin

returns: x(n),y(n)

RLIMIT Truncates the portion of the fractures lying outside of the flow or generation region and recalculates the fracture endpoints and length if needed. Fills the [mask (2,mfrc)] and [kut(mfrc)] arrays.

given: nfrac,nsets,iseti(1,2),maxfrc,frac(nfrac,8),
frac(nfrac,9),frac(nfrac,10),xmesh,ymesh,rotan.

returns: frac(nfrac,2),frac(nfrac,4),frac(nfrac,5),frac(nfrac,6)
frac(nfrac,7),kut(nfrac),mask(2,nfrac)

SPATIA Reads fracture information by subregions and then generates fractures by subregions.

given: maxfrc,dseed,frac(maxfrc,10),icent,itole,
iranf,iunits,ikeep,imesh,nfrc,iplot,
xgene,ygene,xmesh,ymesh,rotan,nsgene, nsets,iseti(20,8)

reads: rseti(i,11),ichar,idist,ev,sd,ycept,slope,
rlambmn,rlbar,nfracn,theta,const

returns: frac(i,j), j = 1,5

UNIFOD Generates random variables distributed uniformly between (center -range) and (center +range)

given: n,dseed,center,range

returns: a(n)

WRENUM Computes boundary conditions, and writes the input decks RENUM%%.DAT for the mesh optimization program RENUM.

given: frac(maxfrc,10),kut(mfrc),knode(mkey),
tint(2,1),ifrac(2,1),xgene,ygene,xmesh,
ymesh,rotan,nsgene,itole,iranf,kenzi,

imesh,nfrac,iplot,nnodes,neleme,iray,
idate,lrenum,visc,spgr,ikeep,transm

reads: [bcode(i),i=1,4],[bvalu(i),i=1,4]

prints: complete mesh specification

Appendix B

FMG - Program Listing

```

c      program fmq
c      ***** s e t u p *****
c
c      parameter (mfr=100000)
c      parameter (mnod=70000)
c      parameter (mkey=90000)
c      common/llles/ lplot,lrenum
c      common/frac/  frac(mfr,10)
c      common/lfrac/ lfrac(2,mnod)
c      common/lnew/  lnw(mfr)
c      common/lold/  lold(mfr)
c      common/lwk/   lwk(mfr)
c      common/kind/  kind(mnod)
c      common/knode/ knode(mkey)
c      byte kind
c      common/kut/   kutone,kut(mfr)
c      common/mask/  mask(2,mfr)
c      integer*2 mask
c      common/mesh/  xgene,ygene,rgene,xmesh,ymesh,rotan,
c      rmesh,xhole,yhole,rhole,nsqgene
c      common/next/  next(mfr)
c      common/param/ itole,lrani,lunits,lkeep,lmesh,lprnt,nfrac,
c      lplot,lqene
c      common/pl/    pl180
c      common/qc/    visc,spqr,qc,itrans
c      common/seti/  nsets,lset1(20,8),rset1(20,11)
c      common/stud/  lstud,astud(10),ystud(10)
c      common/tint/  tint(2,mnod)
c      common/title/ title,lray,ldate
c      common/xy/    xy(mfr,6)
c      integer*2 lside(mfr)
c      dimension nfrs(4)
c      character*80 title(2)
c      character*10 faccs(2)
c      character*9 ldate,odate
c      character*19 lray,oray
c      character*3 cqsms(2),fatat(2)
c      double precision dseed
c      data cqsms/'cgs','mks'/
c      data faccs,fatatt/'sequential','append','new','old'/
c      data lplot,lrenum/-2,0/
c
c      define pl180.
c
c      pl180 = atan(1.)/45.
c
c      *** open files
c
c      open (unit=1,readonly,file='fmq.inp',status='old')
c      open (unit=6,file='fmq.out',status='unknown')
c      open (unit=15,readonly,file='plot.dat',status='old')
c      lstud=0
c      open (unit=20,readonly,file='study.dat',status='old',err=19)
c      lstud=1
c
c      *** set maxfr & mkey to dimension the whole program
c
c      19 maxfr= mfr
c      maxnod= mnod
c      maxkey= mkey
c
c      *** request job information.
c
c      lray = 'fmgrun Version 1.00'

```

```

c      call date (ldate)
c
c      ***** i n p u t *****
c
c      *** read control variables
c
c      read (1,260) lcont,lplot,lmesh,lprnt
c      read (1,260) lkeep,lunits,lqene
c      read (1,270) lrani,dseed
c      write (6,410) cqsms(lunits+1)
c      write (6,290) lcont,lplot,lmesh
c
c      if (lunits.eq.0) then
c         visc=.01
c         spqr=980.66
c      else
c         visc=.001
c         spqr=9806.6
c
c      endif
c      qc=spqr/(12.*visc)
c
c      if (lstud.ne.0) read(20,'(12)') lstud
c      if (lstud.ne.0) then
c         do 1130 i=1,lstud
c            read(20,1140) astud(i),ystud(i)
c1140      format (2f6.2)
c1130      continue
c         endif
c         if (lcont.eq.4) go to 49
c         if (lcont.eq.3) then
c            stop ' option lcont=3 is no longer valid'
c         endif
c
c      *** read input variables
c
c      read (1,330) title
c      if (lqene.eq.0) then
c         read (1,250) xgene,ygene,nsqgene
c      else
c         read(1,250) rgene,dumy,nsqgene
c         xgene=2.*rgene
c         ygene=2.*rgene
c      endif
c      read (1,280) nsets,ltolc
c
c      ***** f r a c t u r e   g e n e r a t i o n *****
c
c      *** call subroutine fragen to read in or generate fracture
c      *** characteristics and store them in array frac
c
c      call fragen (maxfr,dseed,frac)
c      *** program stop if nfrac greater than maxfr
c
c      if (nfrac.gt.maxfr) then
c         write(6,140) nfrac,maxfr
c         stop
c      endif
c
c      *** print the contents of frac.
c

```

```

        write(6,210) iray,ldate
        write(6,450)
        write(6,470) nfrac
        do 30 m=1,nsets
            write(6,220)
            write(6,230) m,iset1(m,2),rset1(m,10)
            write(6,240)
        30 continue
c
c *** fill array frac ---
c *** 1. call subroutine eqline to calculate coefficients of the
c *** equation of the line on which the fracture lies,
c *** 2. call subroutine endpts to calculate fracture endpoints,
c
        call eqline(maxfrc,frac)
        call endpts(maxfrc,frac)
c
c *** calculate and print fracture statistics for each set
c
        write(6,540)
        call pfs (maxfrc,frac,xy)
c
c *** 3. call subroutine rlimit or climt to truncate any fractures
c *** falling outside of the generation block.
c
        if (igene.eq.0) then
            rotan=0.
            xmesh=xgene
            ymesh=ygene
            call rlimit (icont,maxfrc,frac)
        else
            rmesh=rgene
            rhole=0.
            xhole=2.*rmesh
            yhole=2.*rmesh
            call climt (icont,maxfrc,frac)
        endif
c
c *** print the contents of frac and kut.
c
        write(6,210) iray,ldate
        write(6,480)
        write(6,460) xgene,ygene,ngene**2
        write(6,470) nfrac
c
c *** write frac and kut to tape5.
c
        open (unit=5,file='frac.txt',status='unknown')
        write (5,300) iray,ldate
        write (5,330) title
        write (5,420) nsets
        write (5,350) ((iset1(i,j),j=1,8),i=1,nsets)
        write (5,370) ((rset1(i,j),j=1,10),i=1,nsets)
        write (5,430) nfrac,xgene,ygene
        write (5,380) ((frac(i,j),j=1,12),kut(i),i=1,nfrac)
        close (unit=5)
c
c *** calculate and print fracture statistics for each set
c
        write(6,550)
        call pfs (maxfrc,frac,xy)
c
c *** write data to unit 8

```

```

c
        open (unit=8,file='frac.dat',status='unknown',form='unformatted')
        write (8) iray,ldate,title
        if (igene.eq.0) then
            write (8) nsets,nfrac,xgene,ygene
        else
            write (8) nsets,nfrac,rgene
        endif
        write (8) ((iset1(i,j),j=1,8),i=1,nsets),
        1 ((rset1(i,j),j=1,10),i=1,nsets)
        do 11=1,nfrac,1000
            12=min0(nfrac,11+999)
            write (8) ((frac(i,j),j=1,10),kut(i),i=1,12)
        enddo
        close (unit=8)
c
c *** if iplot = 1, write an input file for a plotting program on tape3
c
        if (iplot.ge.1.and.icont.ne.5) call plococ(maxfrc,frac)
c
        if (icont.eq.1) then
            write(6,150)
            stop
        endif
c
c *** open data in unit 8
c
        49 open (unit=8,file='frac.dat',status='old',readonly,form='unformatted')
c
c ***** f l o w   r e g i o n *****
c
        50 if (igene.eq.0) then
            read (1,440,end=130) xmesh,ymesh,rotan
        else
            read (1,445,end=130) rmesh,rhole,xhole,yhole
            xmesh=rmesh*2.
            ymesh=rmesh*2.
        endif
c
c *** read data from unit 8
c
        read (8) oray,odate,title
        if (igene.eq.0) then
            read (8) nsets,nfrac,xgene,ygene
        else
            read (8) nsets,nfrac,rgene
        endif
        read (8) ((iset1(i,j),j=1,8),i=1,nsets),
        1 ((rset1(i,j),j=1,10),i=1,nsets)
        do 11=1,nfrac,1000
            12=min0(nfrac,11+999)
            read (8) ((frac(i,j),j=1,10),kut(i),i=1,12)
        enddo
        rewind (unit=8)
c
        if (xmesh.ne.xgene).or.(ymesh.ne.ygene).or.(rotan.ne.0.).or.
        ' (rmesh.ne.rgene).or.(rhole.ne.0.) then
c
c *** call subroutine rlimit or climt to eliminate or truncate
c *** fractures falling outside of the flow region.
c
        if (igene.eq.0) then

```

```

        call rlimit (icont,maxfrc,frac)
      else
        call climit (icont,maxfrc,frac)
      endif
    endif
    if (nfrac.eq.0) write(6,170) rotn
c
c *** print the contents of frac and kut.
c
    write(6,210) iray,ldate
    write(6,490)
    write(6,500)
    write(6,510) amesh,ymesh,rotn
    write(6,530) nfrac
c
c *** calculate and print fracture statistics for each set
c
    write(6,560)
    call pfs (maxfrc,frac,xy)
c
c
c *** if iplot = 1, write an input file for a plotting program on tape3
c
c
    if (iplot.ge.1.and.nfrac.ne.0.and.rotn.eq.0..and.icont.ne.5) then
      call plococ(maxfrc,frac)
    else
      if(iplot.lt.0)iplot=0
    endif
    call connec(icont,maxfrc,inside,frac)
    if (icont.eq.5)goto 50
    call wrenum(maxfrc,frac)
c
c *** if iplot = 2, write an input file for a plotting program on tape3
c
    if (iplot.eq.2.and.nfrac.ne.0) call plococ(maxfrc,frac)
    go to 50
c
130 write(6,160)
    stop
c
c ***** format statements *****
c
c
140 format('1',39('-'))/
    1 ' program stop,nfrac is greater than maxfrc,nfrac = ',16/
    2 ' maxfrc= ',16/1x,55('-')
150 format('0normal program stop,icont=1')
160 format('0normal program stop,end of input')
170 format('0no fractures in flow region for rotn=',f6.2)
210 format('0',a19,' - ',a9)
220 format('0',10(/),6x,'set      number of fractures',6x,
    1 'density of fractures'/)
230 format('0',5x,15,15x,15,15x,1p,e10.3)
240 format(10(/))
250 format(2(10x,f10.4),10x,15)
260 format(10x,15,3(15x,15))
270 format(10x,15,15x,d15.8)
280 format(3(10x,15))
290 format('0icont= ',15,10x,'iplot = ',15,10x,'imesh = ',15)
300 format(a19,' - ',a9)
320 format(a)
350 format(8110)

```

```

370 format(1p,5e12.4)
380 format(2f10.4,1p,e10.3,0p,4f10.2,3x,z2)
410 format('*** units are ',a,' ***')
420 format('---nsets---',15)
430 format('---nfrac---',15,'---xgene--- ',f10.4,'---ygene---',f10.4)
440 format(3(10x,f10.4),7x,11)
445 format(4(10x,f10.4))
450 format('0fracture generation')
460 format('0the size of the generation region is ',f8.1,' by ',f8.1/
    1 ' the number of subregions is ',13)
470 format('0the number of fractures generated or read in is ',15)
480 format('0truncated fractures of',
    1 ' generation stage')
490 format('0fractures in flow region')
500 format(' (all fractures included)')
510 format('0the size of the flow region is ',f8.1,' by ',f8.1/
    1 ' the angle of rotation is ',f6.2)
530 format('0the number of fractures in the flow region is ',15)
540 format('0fracture statistics')
550 format('0fracture statistics'/
    1 ' of truncated fractures')
560 format('0fracture statistics'/
    1 ' of fractures in flow region')
    end

```

```

      subroutine circxy (x,y,n,dseed,rs,itole)
c
c *** this subroutine generates random fracture centers
c *** in a circle with radius rs
c
      dimension x(n),y(n)
      double precision dseed
c
      toler=10.**itole
      rt=2*rs*toler
      do i=1,n
10    a=ggubfs(dseed)-0.5
        b=ggubfs(dseed)-0.5
        if (a*a+b*b.gt.0.25) goto 10
        x(i)=float (int (rt*a))/toler
        y(i)=float (int (rt*b))/toler
      enddo
c
      return
      end

```

```

subroutine cllmt (lcont,maxfr,frac)
parameter (nc=10)
common/param/ ltole,lrant,iunits,lkeep,lmesh,lprnt,nfrac,
               lplot,lgene
common/lnew/   lnw(100)
common/lold/   lold(100)
common/kut/    kutone,kut(100)
common/mask/   mask(2,100)
integer*2 mask
common/mesh/   xgene,ygene,xmesh,ymesh,rotan,
               rmesh,xhole,yhole,rhole,nsgene
common/next/   next(100)
common/pi/     pi180
common/set1/   nsets,iset1(20,8),rset1(20,11)

c
c
c *****
c
c this subroutine truncates a fracture if one or both of its
c endpoints fall outside of the circle of radius rmesh,
c and discards the fracture if it is outside the block.
c
c if the fracture is truncated, the coordinate of the endpoints
c and the length of the fracture are recalculated.
c
c nfrac is the number of fractures.
c
c one components of frac may be recalculated in this subroutine -
c   frac(1,2) = length,
c   frac(1,4) = x1, x coordinate of endpoint 1,
c   frac(1,5) = y1, y coordinate of endpoint 1,
c   frac(1,6) = x2, x coordinate of endpoint 2,
c   frac(1,7) = y2, y coordinate of endpoint 2.
c
c the components of frac used in this subroutine are ---
c   frac(1, 8) = a = -sin of orientation,
c   frac(1, 9) = b = cos of orientation,
c   frac(1,10) = c,
c where a, b, and c are the coefficients in the equation of the
c line, ax + by + c = 0, on which fracture i lies.
c
c   kut(i) = truncation code.
c   next(i) = where the next truncated fracture sits
c
c dimension frac(maxfr,10)
c dimension t(4),is(4)
c
c toler=10.**-ltole
c k=1
c n2=0
c lnext=0
c rmsq=rmesh*rmesh
c rn2=rmesh*2.
c rhsq=rhole*rhole
c
c do 170 i=1,nsets
c
c   m=0
c
c   n1=n2+1
c   n2=n1+iset1(i,2)-1
c   if (n2.lt.n1) go to 170
c
c   l=n1

```

```

do while (l.le.n2)
c
c calculate intersection with outer disc
c
   l160=0
   next(i)=0
   x1=frac(i,4)
   y1=frac(i,5)
   x2=frac(i,6)
   y2=frac(i,7)
   alen=frac(i,2)
   ku=0
   a=alen*alen
   b=x1*(x2-x1)+y1*(y2-y1)
   c=x1*x1+y1*y1-rmsq
   delt=b*b-a*c
   if(delt.le.0.)goto 150
   delt=sqrt(delt)
   t1=(-b-delt)/a
   if(t1.ge.1.)goto 150
   t2=(-b+delt)/a
   if(t2.le.0.)goto 150
   if(t1.gt.0.)then
     x1=x1+t1*(x2-x1)
     y1=y1+t1*(y2-y1)
     t2=(t2-t1)/(1.-t1)
     alen=alen*(1.-t1)
     if(alen.lt.toler)goto 150
     ku=ku+40
   endif
   if(t2.lt.1.)then
     x2=x1+t2*(x2-x1)
     y2=y1+t2*(y2-y1)
     alen=alen*t2
     if(alen.lt.toler)goto 150
     ku=ku+3
   endif
c
c calculate intersection with inner disc
c
   a=alen*alen
   xx1=x1-xhole
   yy1=y1-yhole
   xx2=x2-xhole
   yy2=y2-yhole
   b=xx1*(xx2-xx1)+yy1*(yy2-yy1)
   c=xx1*xx1+yy1*yy1-rhsq
   delt=b*b-a*c
   if(delt.le.0.)goto 140
   delt=sqrt(delt)
   t1=(-b-delt)/a
   if(t1.ge.1.)goto 140
   t2=(-b+delt)/a
   if(t2.le.0.)goto 140
   if(t1.lt.0..and.t2.gt.1.)goto 150
   if(t1.gt.0.)then
     ku2=ku-16*(ku/16)
     kuplus=1-ku2
     if(t2.lt.1.)then
       frac(i,4)=x1+(t1+t2)*(x2-x1)/2.
       frac(i,5)=y1+(t1+t2)*(y2-y1)/2.
       if (ku2.ne.0)then
         frac(i,2)=sqrt( (frac(i,6)-frac(i,4))**2 +
                        (frac(i,7)-frac(i,5))**2 )

```

```

c
if (n2.lt.n1) go to 170
l=n1

```

```

frac(1,5)=y1+(t1+t2)* (y2-y1)/2.
if (ku2.ne.0) then
frac(1,2)=sqrt( (frac(1,6)-frac(1,4))**2 +
(frac(1,7)-frac(1,5))**2 )

```

```

else
frac(1,2)=alen*(1.-(t1+t2)/2.)
frac(1,6)=x2
frac(1,7)=y2
endif
x2=x1+t1*(x2-x1)
y2=y1+t1*(y2-y1)
alen=alen*t1
ku=ku*kuplus
l160=l
if (k.eq.1) call move(1,nfrac,n2,frac,maxfrac)
goto 140
endif
x2=x1+t1*(x2-x1)
y2=y1+t1*(y2-y1)
alen=alen*t1
ku=ku*kuplus
endif
if (t2.lt.1.) then
kul=ku/16
kuplus=16*(1-kul)
x1=x1+t2*(x2-x1)
y1=y1+t2*(y2-y1)
alen=alen*(1.-t2)
ku=ku*kuplus
endif

```

```

c *** store information about fractures part or all of which fall into
c *** the flow region.

```

```

c
140 ix1=(x1+rmesh)*nc/rm2
ix2=(x2+rmesh)*nc/rm2
iy1=(y1+rmesh)*nc/rm2
iy2=(y2+rmesh)*nc/rm2
ix=min(ix1,ix2)
iy=min(iy1,iy2)
ix2=ix1+ix2-ix+1
iy2=iy1+iy2-iy+1
ix2=2**min(ix2,nc)
iy2=2**min(iy2,nc)
ix=max(ix,0)
iy=max(iy,0)
ix1=2**min(nc-1,ix)
iy1=2**min(nc-1,iy)
mask(1,k)=ix2-ix1
mask(2,k)=iy2-iy1
kut(k)=ku
frac(k,1)=frac(1,1)
frac(k,2)=alen
frac(k,3)=frac(1,3)
frac(k,4)=x1
frac(k,5)=y1
frac(k,6)=x2
frac(k,7)=y2
do j=8,10
frac(k,j)=frac(1,j)
enddo

```

```

c
next(k)=0
k16=kut(k)/16
if (k16.eq.1.or.kut(k)-16*k16.eq.1) then
next(k)=inext
inext=k

```

```

k=k+1
m=m+1
150 if (l160.eq.0) l=l+1
c
c
c
c
c
170 continue
c
nfrac=k-1
k16=kut(nfrac)/16
if (k16.ne.1.and.kut(nfrac)-16*k16.ne.1) next(nfrac)=inext
c
return
end

```

```

      subroutine connec(icont,maxfr,inside,frac)
c
c   this subroutine sets up intersection information
c   and uses the following arrays
c   frac - array of fracture information
c   kut - which sides are crossed by this fracture
c   next - next fracture which crosses a side
c
c   during processing next and kut are destroyed
c   the arrays created are
c   next - array of initial fracture (negative means more than one)
c   kut - index into arrays tint and ifrac
c   tint - t value of intersection
c   ifrac- other fracture in intersection
c
      common/param/ itole,iranf,iunits,ikeep,imesh,iprnt,nfrac,
      , iplot,igene
      common/mesh/ xgene,ygene,rgene,xmesh,ymesh,rotan,
      , rmesh,xhole,yhole,rhole,nsgene
      common/kut/ kutone,kut(1)
      common/mask/ mask(2,1)
      integer*2 mask,mask1,mask2
      common/next/ next(1)
      common/knode/ knode(1)
      common/old/ old(1)
      common/new/ new(1)
      common/ifrac/ ifrac(2,1)
      common/line1/ nnodes,neleme
      common/kind/ kind(1)
      byte kind
      common/tint/ tint(2,1)
      common/pi/ pi180
c
      dimension frac(maxfr,10)
      dimension is(2)
      integer*2 inside(maxfr)
      OPEN (99,status='unknown')
c
c   this is to make it unnecessary to move fracture informaton
c
      do i=1,nfrac
        old(1)=i
        new(1)=i
        inside(1)=kut(1)
      enddo
c
      toler=10.**(-itole-1)
      inext=nfrac
      k=1
      if (icont.lt.5.and.igene.eq.0)then
        if (inside(old(nfrac)).eq.0) inext=next(nfrac)
      else
        k16=inside(old(nfrac))/16
        if (k16.ne.1.and.inside(old(nfrac))-16*k16.ne.1)inext=next(nfrac)
      endif
c
      do while (inext.ne.0)
        if (icont.ne.5.and.igene.eq.0)then
          do while (inside(old(k)).ne.0.and.k.lt.inext)
            k=k+1
            next(k)=0
          enddo
        else
          do while ((inside(old(k))/16.eq.1.or.

```

```

      , mod(inside(old(k)),16).eq.1)
      , .and.k.lt.inext)
        k=k+1
        next(k)=0
      enddo
    endif
    i=next
    inext=0
    if (i.gt.k) then
      inew(k)=i
      iold(i)=k
      inew(1)=k
      iold(k)=i
      inext=next(i)
      next(i)=0
      next(k)=0
    endif
    enddo
    next(nfrac)=0
    if (icont.ne.5.and.igene.ne.1)then
      if (inside(old(k)).eq.0) k=k-1
    else
      k16=inside(old(k))/16
      if (k16.ne.1.and.inside(old(k))-16*k16.ne.1)k=k-1
    endif
c
    if (k.eq.0) then
      if (ikeep.ne.1) then
        nfrac=0
        go to 60
      endif
      k=1
    endif
    kutone=0
    nnodes=0
    nvoid=0
    ivoid=0
    neleme=0
    nextra=0
    indint=0
    ilevel=0
    ifrc1=1
    ifrc2=k
    k1=0
    k2=1
10 ifrc3=ifrc2+1
    k0=ifrc3
    do i1=ifrc1,ifrc2
      inext=0
      k1=k1+1
      i=k2
      if (i.lt.ifrc1) then
        ifrc3=ifrc3-1
        i=ifrc3
      endif
      io=old(i)
      k2=k1+1
      int1=indint+1
      ifrc=next(i)
      next(i)=0
      next(k1)=0
      if (ilevel.eq.0.or.igene.eq.1) then
c
c   include intersections with a side

```



```
do while ((iside(loid(k))/16.eq.1.or.
```

```
c Include intersections with a side
```

```
c
is(1)=iside(1o)/16
is(2)=iside(1o)-16*is(1)
if (icont.eq.5.and.(is(1).eq.3.or.is(2).eq.3))goto 25
do j=1,2
  if (is(j).ne.0) then
    indint=indint+1
    nnodes=nnodes+1
    knode(indint)=nnodes
    ifrac(1,nnodes)=k1
    ifrac(2,nnodes)=is(j)
    tinc(1,nnodes)=(3*j-4)*frac(1o,2)
    tinc(2,nnodes)=0.
    kind(nnodes)=0
  endif
enddo
endif
c
c Include intersections with previous fractures
c
do while (jfrc.ne.0)
  jint1=kut(jfrc-1)+1
  jint2=kut(jfrc)
  do while (knode(jint2).eq.0)
    jint2=jint2+1
  enddo
  do jj=jint1,jint2
    j=knode(jj)
    if (ifrac(1,j).eq.1) go to 20
  enddo
  stop ' sync lost - jfrc points to fracture w/o backpointer'
20 indint=indint+1
  knode(indint)=j
  j1=jfrc
  jfrc=ifrac(2,j)
  ifrac(1,j)=j1
  ifrac(2,j)=k1
enddo
c
c check for intersections with fractures not previously included
c
jfrc=max0(k1+1,ifrac1)
if (jfrc.le.nfrac) then
  ai=frac(1o,8)
  bi=frac(1o,9)
  ci=frac(1o,10)
  x1i=frac(1o,4)
  y1i=frac(1o,5)
  x2i=frac(1o,6)
  y2i=frac(1o,7)
  msk1=mask(1,1o)
  msk2=mask(2,1o)
c
  j1=jfrc-1
  nfrac=nfrac
  do 30 jj=jfrc,nfrac
    j1=j1+1
    if (j1.ge.1fr3.and.j1.lt.k0) go to 30
    j=j1
    jo=loid(j1)
    if (frac(1o,1).eq.frac(jo,1)) GOTO 30
    if ((mask(1,jo).and.msk1).eq.0) go to 30
```

```

b)=frac(jo,9)
c)=frac(jo,10)
c
c check to see if (x1i,y1i) and (x2i,y2i) are on different sides of
c the line of the jth fracture.
c
d1i = aj*x1i + bj*y1i + ci
d2i = aj*x2i + bj*y2i + ci
if (d1i*d2i.gt.0) go to 30
x1j=frac(jo,4)
y1j=frac(jo,5)
x2j=frac(jo,6)
y2j=frac(jo,7)
c
c check to see if (x1j,y1j) and (x2j,y2j) are on different sides of
c the line of the 1th fracture.
c
d1j = ai*x1j + bi*y1j + ci
d2j = ai*x2j + bi*y2j + ci
if (d1j*d2j.gt.0) go to 30
c
c they intersect
c
d1i=abs(d1i)
d2i=abs(d2i)
denom1=d1i+d2i
d1j=abs(d1j)
d2j=abs(d2j)
denomj=d1j+d2j
if (denom1*denomj.eq.0) go to 30
if (icont.eq.5)then
  if (iside(jo)/16.eq.3.or.mod(iside(jo),16).eq.3)then
25 ncon=0
    open(unit=50,file='connections',status='unknown')
    read(50,*,end=26)ncon
26 ncon=ncon+1
    rewind(unit=50)
    write(50,*)ncon
    close (unit=50)
    goto 60
  endif
endif
if (j.gt.k) then
  if (ifrc3.lt.k0.and.1+1.lt.k0) then
    k0=k0+1
    loid(k0)=jo
    lnew(jo)=k0
    ko=loid(nfrac)
    loid(j1)=ko
    lnew(ko)=j1
    j1=j1-1
    j=k0
    nfrac=nfrac-1
  else
    k=k+1
    ko=loid(k)
    loid(k)=jo
    lnew(jo)=k
    loid(j1)=ko
    lnew(ko)=j1
    j=k
  endif
```

```

j=j1
jo=old(j1)
IF (frac(1,1).EQ.frac(jo,1)) GOTO 30
IF ((mask(1,jo).and.mask1).eq.0) go to 30

```

```

old(j1)=ko
lnew(ko)=j1
j=k
endif

```

```

30 IF (ivoid.ne.0) then
    nvoid=nvoid-1
    jnode=ivoid
    ivoid=ifrac(1,ivoid)
    else
        nnodes=nnodes+1
        jnode=nnodes
    endif
    indint=indint+1
    knode(indint)=jnode
    ifrac(1,jnode)=j
    ifrac(2,jnode)=next(j)
    tint(1,jnode)=d1+frac(io,2)/denom1
    tint(2,jnode)=d1+frac(jo,2)/denom2
    kind(jnode)=0
    next(j)=k1
    continue
endif
neleme=neleme+indint-int1
IF (indint.eq.int1) then
    if (ikeep.eq.0) then
        lnext=0
        indint=indint-1
        l=int1
        lnd=k1
        k1=k1-1
        ifrc3=ifrc3-1
        k2=ifrc3
40 jnode=knode(l)
        knode(l)=0
        jfr=ieor(lnd,ieor(ifrac(1,jnode),ifrac(2,jnode)))
        IF (jnode.eq.nnodes) then
            nnodes=nnodes-1
        else
            nvoid=nvoid+1
            ifrac(1,jnode)=ivoid
            ivoid=jnode
            kind(jnode)=1
        endif
        if (jfr.gt.0) then
            neleme=neleme-1
            jint2=kut(jfr)
            jint1=kut(jfr-1)+1
            l=jint1-1
            do j=jint1,jint2
                lnode=knode(j)
                if (lnode.eq.jnode) then
                    knode(j)=0
                elseif (lnode.ne.0) then
                    l=l+1
                    knode(j)=0
                    knode(l)=lnode
                endif
            enddo
            lnd=jfr
            if (l.le.jint1) go to 40
        endif
        elseif (int1.lt.indint) then
c
c      sort elements by ascending tint's
c
        i2=indint
        j1=int1+1

```

```

do while (i2.ge.j1)
    j2=i2
    i0=1
    i1=0
    i2=0
    j2=knode(j1-1)
    kj2=(k1-ifrac(1,j2))/(ifrac(2,j2)-ifrac(1,j2))
    if ((kj2.and..not.1).ne.0) stop 'kj2 error 1'
    do j=j1,i2
        j1=j2
        kj1=kj2
        j2=knode(j)
        kj2=(k1-ifrac(1,j2))/(ifrac(2,j2)-ifrac(1,j2))
        if ((kj2.and..not.1).ne.0) stop 'kj2 error 2'
        if (tint(kj2+1,j2).lt.tint(kj1+1,j1)) then
            knode(j)=j2
            knode(j1)=j1
            j2=j1
            kj2=kj1
            i2=j-1
            i1=i1+10*i2
            i0=0
        endif
    enddo
    j1=max0(i1,int1+1)
    enddo
    if (i.ne.k1) then
        iold(i)=0
        iold(k1)=io
        inew(io)=k1
    endif
    endif
    kut(k1)=indint
    if (int1.gt.indint) then
        nexta=nexta+2
    else
        inode=knode(int1)
        kj2=(k1-ifrac(1,inode))/(ifrac(2,inode)-ifrac(1,inode))
        if ((kj2.and..not.1).ne.0) stop 'kj2 error 3'
        if (tint(kj2+1,inode).gt.toler) then
            nexta=nexta+1
        elseif (tint(kj2+1,inode).lt.0.) then
            tint(kj2+1,inode)=0.
        endif
        inode=knode(indint)
        kj2=(k1-ifrac(1,inode))/(ifrac(2,inode)-ifrac(1,inode))
        if ((kj2.and..not.1).ne.0) stop 'kj2 error 4'
        if (tint(kj2+1,inode).lt.frac(io,2)-toler) then
            nexta=nexta+1
        elseif (tint(kj2+1,inode).gt.frac(io,2)) then
            tint(kj2+1,inode)=frac(io,2)
        endif
    endif
    enddo
    llevel=llevel+1
    if (k.ge.k0) then
        ifrc1=k0
        ifrc2=k
        go to 10
    else
        if (ikeep.ge.2.and.k.lt.nfrac) then
            k=k+1
            ifrc1=k
            ifrc2=k

```

```

        go to 10
      endif
      nfrac=k1
    endif
c
    nnodes=nnodes+nextra-nvoid
    neleme=neleme+nextra
c
    if (lkeep.ne.0) return
    nnodes=nnodes-nextra
    neleme=neleme-nextra
c
60 return
end

```

```

      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
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 expnod (a,n,dseed,ev)
   return
40 stop 'gamma distribution disabled'
c   call gammad (a,n,dseed,ev,sd)
c   return
50 call unifod (a,n,dseed,ev,sd)
   return
end

```

```

      subroutine endpts(maxifrc,frac)
      common/param/ ltole,lfanf,lunits,lkeep,lmesh,lprnt,nfrac,
      , lplot,lgenw
c
c
c .....
c
c this subroutine calculates the coordinates of the endpoints of a
c fracture given the center, orientation and length of the fracture.
c
c nfrac is the number of fractures.
c
c the components of frac used in this subroutine are ---
c   frac(1,1) = orientation,
c   frac(1,2) = length,
c   frac(1,4) = xc, x coordinate of fracture center,
c   frac(1,5) = yc, y coordinate of fracture center.
c
c the components of frac that are calculated in this subroutine
c are ---
c   frac(1,4) = x1, x coordinate of endpoint 1,
c   frac(1,5) = y1, y coordinate of endpoint 1,
c   frac(1,6) = x2, x coordinate of endpoint 2,
c   frac(1,7) = y2, y coordinate of endpoint 2.
c
c the endpoint (x1,y1) lies on the ray which forms an angle =
c frac(1,1) with the positive x-axis. (x2,y2) lies on the ray
c forming an angle = frac(1,1) + 180 with the positive x-axis.
c
c .....
c
c dimension frac(maxifrc,10)
c
c do 110 i=1,nfrac
c
c   calculate coordinates of endpoints
c
c     hien=.5*frac(1,2)
c     dx=hien*frac(1,9)
c     dy=hien*frac(1,8)
c     xc=frac(1,4)
c     yc=frac(1,5)
c     frac(1,4)=xc-dx
c     frac(1,5)=yc-dy
c     frac(1,6)=xc+dx
c     frac(1,7)=yc+dy
c
c 110 continue
c
c   return
c   end

```

```

subroutine eqline (maxfrc,frac)
common/param/ itole,lrani,lunits,ikeep,imesh,iprint,nfrac,
,          iplot,lgene
common/pi/      pi180

c
c          version 1.1 (oct 1981)
c          .....
c
c  this subroutine calculates the coefficients, a, b and c, of
c  the line through the fracture center with the given orientation.
c
c  nfrac is the number of fractures.
c
c  the general form of the line is ax + by + c = 0.
c  a, b and c are stored in the array frac ---
c      frac(1,8) = a,
c      frac(1,9) = b,
c      frac(1,10) = c.
c
c  the components of frac used in this subroutine are ---
c      frac(1,1) = orientation,
c      frac(1,4) = xc, x coordinate on fracture,
c      frac(1,5) = yc, y coordinate on fracture.
c
c          .....
c
c  dimension frac(maxfrc,10)
c
c  do 140 i=1,nfrac
c
c  set local variables.
c
c      orie=frac(i,1)
c      xc=frac(i,4)
c      yc=frac(i,5)
c
c  convert orie from degrees to radians.
c
c      orie=orie*pi180
c
c  line on which fracture lies is neither horizontal nor vertical
c
c      a=-sin(orie)
c      b=cos(orie)
c
c      frac(i,8)=a
c      frac(i,9)=b
c      frac(i,10)=-b*yc-a*xc
c
c 140 continue
c
c  return
c  end

```

```

      subroutine exponnd (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

```

```

subroutine fragen (maxfrc,dseed,frac)
common/param/ itole,iranf,iunits,ikeep,imesh,iprnt,nfrac,
               lplot,lgene
common/mesh/   xgene,ygene,rgene,xmesh,ymesh,rotan,
               rmesh,ahole,yhole,rhole,nsgene
common/set1/   nsets,iset1(20,8),rset1(20,11)
common/pl/     pl100
common/qc/     vlsq,spqr,qc,ltrans
c
c *****
c
c this subroutine reads in fracture information and then either
c reads in or generates the following fracture characteristics -
c orientation, length, aperture, and the coordinates of the
c fracture center.
c
c the following variables or arrays from the main program are
c used ---
c
c   nsets = number of fracture sets,
c
c   xgene = x dimension of generation region,
c   ygene = y dimension of generation region,
c   nsgene*2 = number of subregions for generation
c
c   frac(1, 1) = orientation,
c   frac(1, 2) = length,
c   frac(1, 3) = aperture,
c   frac(1, 4) = x1, x coordinate of fracture center or end,
c   frac(1, 5) = y1, y coordinate of fracture center or end,
c   frac(1, 6) = x2, x coordinate of fracture end,
c   frac(1, 7) = y2, y coordinate of fracture end,
c   frac(1, 8) = a, coeff. = -sin(orientation)
c   frac(1, 9) = b, coeff. = cos(orientation)
c   frac(1,10) = 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 read or generation are given
c below ---
c
c   icent = 1 - read in all fracture characteristics,
c           2 - generate fracture center coordinates,
c           3 - generate fracture center coordinates in subregions
c
c   idens = 0 - read and use rlamb
c           1 - read nfrac directly
c           2 - calculate rlamb for r1bar & rlambdal
c
c   lpois = 0 - generate fracture centers in whole gen. region
c           1 - generate frctr centers along line at angle theta
c
c   lchar = 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
c   idist = 1 - normal distribution,
c           2 - lognormal distribution,
c           3 - exponential distribution,
c           4 - gamma distribution (DISABLED)
c           5 - uniform distribution.
c
c in addition, the following variables are read in ---
c
c   nfrac = number of fractures,
c
c   rlamb = no. of fractures per square unit,
c
c   theta = angle of poisson generation line
c
c   itole = number of decimal places in fracture center coordinates
c
c   const = see lchar = 2,
c
c   ew = expected value of statistical distribution,
c
c   sd = standard deviation of statistical distribution.
c
c the array iset1 is used to store the values of nfrac, icent, and
c lchar and idist for each set. the array rset1 is used to store
c the values of const, ew and sd for each set.
c
c *****
c
c dimension frac(maxfrc,10)
c
c double precision dseed
c
c *** if iranf = 0, pick a random dseed
c
c   if (iranf .eq. 0) dseed = secnds(0.0) * 100.0
c   write(6,80)dseed
c
c
c *** zero information matrices
c
c   do 20 i=1,nsets
c     do 10 j=1,8
c       iset1(i,j)=0
c       rset1(i,j)=0.
c     10 continue
c     rset1(i,9)=0.
c     20 continue
c
c   open subreg.dat to read in information about subregions
c
c   if(nsgene.ne.0)then
c     open (unit=7,readonly,file='subreg.dat',status='old')
c     endif
c
c   if(ligene.eq.0)then
c     xg2=xgene/2
c     yg2=ygene/2
c     genare=xgene*ygene
c   else
c     xg2=0.
c     yg2=0.
c     genare=pl100*100.*rgene*rgene
c   endif

```



```

n1=1
c
ltrans = 0
do 70 i=1,nsets
c
  read (1,140) icent,ldens,ipols
  if (icent.lt.0) then
    ltrans = 1
    icent = -icent
  end if
  iset1(i,1)=icent
c
  if (icent.eq.1) then
c *** read in all fracture characteristics
c
    read (1,100) nfrac,theta
    iset1(i,2)=nfrac
    n2=nfrac+n1-1
    read (1,150) ((frac(m,j)),j=1,5),m=n1,n2
    do m=n1,n2
      frac(m,4)=frac(m,4)-xg2
      frac(m,5)=frac(m,5)-yg2
    enddo
    n1=n2+1
    iset1(i,3)=1
    iset1(i,5)=1
    iset1(i,7)=1
    rlamb=nfrac/genare
    rset1(i,10)=rlamb
c
  elseif (icent.eq.3) then
    call spatia(maxfrc,dseed,frac,1,n1,n2,ldens)
  else
    if (ldens.ne.0) then
      read (1,100) nfrac,theta
      rlamb=float(nfrac)/genare
    else
      read (1,110) rlamb,theta
      rset1(i,10)=rlamb
      nfrac=int(genare*rlamb)
    endif
c
  if (ipols.eq.1) then
c
    if ((xgene.ne.ygene).and.(ldens.eq.0)) write(6,90)
c
    theta = theta*pi/180
    slcos = cos(theta)
    slsin = sin(theta)
    if (abs(xgene*slsin).le.abs(ygene*slcos)) then
      scan1 = xgene/cos(theta)
    else
      scan1 = ygene/sin(theta)
    endif
    slcos = scan1*slcos
    slsin = scan1*slsin
    if (ldens.eq.0) nfrac=int(scan1*rlamb)
  endif
c
  n2=nfrac+n1-1

```

```

  if (lqgene.eq.0) then
    call rectay (frac(n1,4),frac(n1,5),n2-n1+1,dseed,
    xgene,ygene,ltole,lpols,slcos,slsin)
  else
    call circay (frac(n1,4),frac(n1,5),n2-n1+1,dseed,
    rgene,ltole)
  endif
c
  i=1
  m=-2
  do 60 k=1,3
    i=i+2
    m=m+3
c
    read (1,100) ichar
    iset1(i,1)=ichar
c *** set fracture characteristic to a constant
c
    if (ichar.lt.3) then
      read (1,160) const
      rset1(i,m)=const
      do 50 j=n1,n2
        frac(j,k)=const
      continue
    50
c *** read in code for statistical distribution and read in
c *** statistical parameters
c
    elseif (ichar.eq.3) then
      read (1,120) idist,ev,sd
      iset1(i,1)=idist
      rset1(i,m+1)=ev
      rset1(i,m+2)=sd
      call distrl (idist,frac(n1,k),n2-n1+1,dseed,ev,sd)
    else
      read (1,130) ycepr,slope,sd
      iset1(i,1)=1
      rset1(i,m)=ycepr
      rset1(i,m+1)=slope
      rset1(i,m+2)=sd
      call normdl (frac(n1,k),n2-n1+1,dseed,ycepr,sd,slope,
      frac(n1,2),ichar)
    endif
c
  60 continue
c
  n1=n2+1
  endif
c *** generate locations of fracture centers
c
  70 continue
c
  nfrac=n2
c
  if (icent.ne.3) rset1(i,11)=rset1(i,10)
c
  close subregion file
c
  if (nsgene.ne.0) close (unit=7)

```

```

      else
        scanl = ygene/sin(theta)
      endif
      sicos = scanl*sicos
      slsin = scanl*slsin
      if (ldens.eq.0) nfrac=int(scanl*rlamb)
    endif
  c
  n2=nfrac+n1-1

```

```

70 continue
c
c nfrac=n2
c
c if (lcent.ne.3) rset1(1,11)=rset1(1,10)
c
c close subregion file
c
c if (nsgene.ne.0) close (unit=7)

```

```

c
80 format('0the initial seed used in the random number generator',
1 ' is ',d15.8)
90 format('0xgene.ne.ygene.use nfrac')
100 format(10x,15,15x,f10.4)
110 format(10x,e10.3,10x,f10.4)
120 format(10x,15,15x,f10.4,10x,f10.4)
130 format(3(10x,f10.4))
140 format(3(10x,15))
150 format(5f10.4)
160 format(10x,f10.4)
end

```

```
subroutine frstal(a,b,n,elen,slen,eapt,sapt,cov,crc,ycept,slope,  
1      lchar,xy)  
common/iwk/   iwk(1)  
c  
c      a=frac(*,2)  
c      b=frac(*,3)  
c  
c *** this subroutine calculates the mean and standard deviation.  
c  
c      dimension a(n),b(n),xy(n,6),beta(2),wk(6)  
c  
c      sl=0.  
c  
c      do 110 j=1,..  
c          c=a(j)*b(j)  
c          sl=sl+c  
c          if (a(j).le.0.) write(6,*) j,a(j)  
c          xy(j,1) = a(j)  
c          if (lchar.ne.5) xy(j,1) = alog10(a(j))  
c          xy(j,2) = b(j)  
110      continue  
c      call rllav(xy,n,n,1,0,beta,sumre,lter,lrnk,iwk,wk,lcr)  
c      slope = beta(1)  
c      ycept = beta(2)  
c      if (lcr.eq.33) then  
c          write(6,300)  
c      elseif (lcr.eq.129) then  
c          write(6,301)  
c      endif  
c      continue  
c  
c      d=float(n)  
c      elna=sl/d  
c      cov = (elna - elen * eapt)  
c      if (slen.eq.0..or.sapt.eq.0.) then  
c          crc = 99.  
c      else  
c          crc = cov/(slen*sapt)  
c      endif  
c      return  
c  
c      300 format(' the best fit curve is likely to be nonunique')  
c      301 format(' calculation terminated prematurely due to rounding',  
c          1 ' errors')  
c      end
```

```

      subroutine frstat (a,n,av,sd)
c
c *** this subroutine calculates the mean and standard deviation.
c
c      dimension a(n)
ccc if orientation is being examined use stator instead
      if (k.eq.1) stop ' frstat called with k=1'
c
      s1=0.
      s2=0.
c
      do 110 i=1,n
        c=a(i)
        s1=s1+c
        s2=s2+c*c
110    continue
c
      d=float(n)
      av=s1/d
      sd=sqrt(abs(s2/d-av*av))
c
      return
      end

```

```

FUNCTION qqubfs(dDUM)
DIMENSION R(97)
real*8 ddum
PARAMETER (M1=259200,IA1=7141,IC1=54773,RM1=3.0500247E-6)
PARAMETER (M2=134456,IA2=0121,IC2=20411,RM2=7.4373773E-6)
PARAMETER (M3=243000,IA3=4561,IC3=51349)
DATA IFF /0/
idum=int(ddum)
IF (IDUM.LT.0.OR.IFF.EQ.0) THEN
c
c "quick fix" to make sure that idum is initialized in
c the proper range
c
      do while (idum.gt.ic1)
        idum=mod(idum,ic1)
      enddo
      IFF=1
      IX1=MOD(IC1-IDUM,M1)
      IX1=MOD(IA1*IX1+IC1,M1)
      IX2=MOD(IX1,M2)
      IX1=MOD(IA1*IX1+IC1,M1)
      IX3=MOD(IX1,M3)
      DO 11 J=1,97
        IX1=MOD(IA1*IX1+IC1,M1)
        IX2=MOD(IA2*IX2+IC2,M2)
        R(J)=(FLOAT(IX1)+FLOAT(IX2)*RM2)*RM1
1      CONTINUE
      dDUM=1.d0
END1)
IX1=MOD(IA1*IX1+IC1,M1)
IX2=MOD(IA2*IX2+IC2,M2)
IX3=MOD(IA3*IX3+IC3,M3)
dDUM=(dDUM+IX3)/M3
IF(J.EQ.97.OR.J.LT.1)stop ' seed too big, should be < 54773 '
CALL qqubfs(dDUM)
R(J)=(IX1)+FLOAT(IX2)*RM2)*RM1
RETURN
END

```

```

      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(.40) 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 s40/0./
c
      if (s40.eq.0) s40=sqrt(.40)
      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+ggubfa(dseed)
110       continue
        sn=s40*(sn-12.5)
120     a(i)=exp(sdn*sn+evn)
c
      return
      end

```

```

c      subroutine move(l,nfrac,n2,frac,maxfrac)
c
c      dimension frac(maxfrac,10)
c
c      if (nfrac.eq.maxfrac)stop ' too many fractures split in climit'
      idif=jmin0(nfrac/10+1,maxfrac-nfrac)
      do ifrac=nfrac,1,-1
        do j=1,10
          frac(ifrac+idif,j)=frac(ifrac,j)
        enddo
      enddo
      n2=n2+idif
      nfrac=nfrac+idif
      l=l+idif
      return
c      end

```

```

      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
            sn=sn+ggubfs(dseed)
110          sn=sn*(sn-12.5)
120          a(i)=sd*sn+ev
c
      return
      end

```



```

      subroutine normd1 (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.  Gn = (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+qgubfs(dseed)
          sn=s48*(sn-12.5)
          fl=b(i)
          ev=ycept+slope*fl
          if (ichar.ne.5) ev=ycept+slope*log10(fl)
          a(i)=sd*sn*ev
          if (a(i).lt.emin) a(i)=emin
120        continue
c
      return
      end

```

```

subroutine oriest(d,a,b,n,ev,sd)
parameter (nslice=180)
common/pi/ pi180
c
c basic statistics for orientation distributions.
c
c   a--sin(orie)
c   b--cos(orie)
c
dimension a(n),b(n),d(n)
dimension ssum(nslice),csum(nslice)
if (n.eq.0) then
  ev=0.
  sd=0.
  return
endif
do i=1,nslice
  ssum(i)=0.
  csum(i)=0.
enddo
do i=1,n
  alpha=amod(d(i),360.)/180.
  alpha=(nslice*alpha)/180
  ialpha=alpha
  is1=ialpha/nslice
  is2=(ialpha+1)/nslice
  i1=ialpha-nslice*is1+1
  i2=ialpha-nslice*is2+2
  ssum(i1)=ssum(i1)+(-1)**is1*a(i)*(1.-alpha)
  csum(i1)=csum(i1)-(-1)**is1*b(i)*(1.-alpha)
  ssum(i2)=ssum(i2)+(-1)**is2*a(i)*alpha
  csum(i2)=csum(i2)-(-1)**is2*b(i)*alpha
enddo
u0=0.
v0=0.
do i=1,nslice
  u0=u0+csum(i)
  v0=v0+ssum(i)
enddo
r0=u0*u0+v0*v0
u=u0
v=v0
do i=2,nslice
  u=u-2*csum(i-1)
  v=v-2*ssum(i-1)
  r=u*u+v*v
  if (r.gt.r0) then
    u0=u
    v0=v
    r0=r
  endif
enddo
ev=0.
if (r0.gt.0) ev=atan2(v0,u0)/pi180
ev=amod(ev+225.,180.)/45.
sd=1.-sqrt(r0)/n
return
end

```

```

c
  subroutine pfs (maxfr, frac, xy)
  common/iwk/ iwk(1)
  common/set1/ nsets, iset1(20,8), rset1(20,11)
c
c *** calculate and print fracture statistics for each set
c
c
  dimension frac(maxfr,10),
  1      elen(20), slen(20), eapt(20), sapt(20),
  2      xy(maxfr,6)
  character*11 ola(3)
  character*7 type(5)
  data ola/'orientation','length','aperture'/
  data type/'normal','lognor.','expon.','gamma.','uniform'/
  data ion/0/
c
  write(6,60) nsets
  write(6,160)
  write(6,170) (i,rset1(i,10),rset1(i,11),i=1,nsets)
  do 10 i=1,nsets
    if (iset1(i,7).eq.4.or.iset1(i,7).eq.5) ion=1
  10 continue
  do 30 k=1,3
    write(6,70) ola(k)
    if (ion.eq.1.and.k.eq.3) then
      write(6,90)
    else
      write(6,80)
    endif
c
    n2=0
    do 20 i=1,nsets
      nf=iset1(i,7)
      if (nf.le.0) then
        write(6,50) i,nf
      else
        n1=n2+1
        n2=n1+nf-1
        i=k*2+1
        ichar=iset1(i,1)
        if (ichar.eq.1) then
          write(6,100) i,nf
        elseif (ichar.eq.2) then
          m=k*3-2
          write(6,110) i,nf,rset1(i,m)
        else
          m=k*3-2
          idist=iset1(i,1+1)
          if (k.eq.1) then
            call orlist (frac(n1,1),frac(n1,8),frac(n1,9),
            1      n2-n1+1,ev,sd)
          else
            call frstat (frac(n1,k),n2-n1+1,ev,sd)
            if (k.eq.2) then
              elen(i) = ev
              slen(i) = sd
            elseif (k.eq.3) then
              eapt(i) = ev
              sapt(i) = sd
            endif
          endif
          if ((ichar.eq.4).or.(ichar.eq.5)) then
            call frstat (frac(n1,2),frac(n1,3),n2-n1+1,

```

```

1      elen(i),slen(i),eapt(i),sapt(i),
2      cov,crc,ycept,slope,ichar,xy)
  write(6,130) i,nf,cov,crc,rset1(i,8),rset1(i,7),
  slope,ycept
  else
    write(6,120) i,nf,type(idist),rset1(i,m+1),rset1(i,m+2),
    ev,sd
  endif
c
  endif
  endif
  20 continue
  30 continue
  if (ion.eq.1) then
    write(6,80)
    do 40 i=1,nsets
      write(6,140) i,eapt(i),sapt(i)
    40 continue
  endif
  write(6,150)
  return
c
  50 format(1x,13,4x,15)
  60 format('0the number of sets is',i3)
  70 format('0// the following table contains statistical',
  1 ' information on the fracture ',a,' of each set')
  80 format('0set no. of meth. of const. type of ev',8x,'sd',
  1 8x,'ev',8x,'sd'/6x,'frac gen. value dist. ',
  2 2(' read in'),1x,2(' computed'))
  90 format(' -- correlation between length and aperture//
  1 '0set no. of meth. of covari correl. 1/n',7x,'c',
  2 9x,'1/n',7x,'c'/6x,'frac gen. ance coef. ',
  3 2(' read in'),1x,2(' computed'))
  100 format(1x,13,4x,15,' read in')
  110 format(1x,13,4x,15,' set to',4x,f8.4)
  120 format(1x,13,4x,15,' generated',11x,a,4(1x,1p,e9.2e2))
  130 format(1x,13,4x,15,' generated',1p,e9.2e2,0p,2x,f7.4,
  1 1p,4(1x,e9.2))
  140 format(1x,13,12x,'same as above',12x,' n/a n/a ',
  1 1p,2(1x,e9.2))
  150 format(10//)
  160 format('0density for set calculated global')
  170 format(13x,13,5x,f10.6,5x,f10.6)
  end

```

```

      eapt(i) = ev
      sapt(i) = sd
    endif
  endif
  if ((lchar.eq.4).or.(lchar.eq.5)) then
    call frstal (frac(n1,2),frac(n1,3),n2-n1+1,

```

```

      subroutine plococ(maxfr,frac)
      common/param/ ltole,lrani, lunits, lkeep, lmesh, lprnt, nfrac,
      , lplot, lgene
      common/mesh/ xgene, ygene, rgene, xmesh, ymesh, rotan,
      , rmesh, xhole, yhole, rhole, nsgene
      common/stud/ istud, xstud(10), ystud(10)
      common/pl/ pl180
      common/files/ lplot, lrenum
      common/title/ title, lray, idate
      character*80 title(2)
      character*11 filer, filer
      dimension corner(2,5)
      dimension frac(maxfr,10)
      data filer, filer/ 'linesgr.dat', 'renumgr.dat' /
c
      lplot=lplot+1
      if (lplot.ge.0) write (filer(6:7),10) lplot
10 format (i2.2)
      open (unit=3, file=filer, status='unknown',
      1 carriagecontrol='list')
c
c draw study regions
c
      if (istud>lplot.gt.0) then
        cosr=cos(rotan*pi/180)
        sinr=sin(rotan*pi/180)
c
        do i=1, istud
          corner(1,4)=cosr*xstud(i)/2 - sinr*ystud(i)/2
          corner(2,4)=cosr*ystud(i)/2 + sinr*xstud(i)/2
          corner(1,2)=corner(1,4)
          corner(2,2)=corner(2,4)
          corner(1,1)=cosr*xstud(i)/2 + sinr*ystud(i)/2
          corner(2,1)=cosr*ystud(i)/2 - sinr*xstud(i)/2
          corner(1,3)=corner(1,1)
          corner(2,3)=corner(2,1)
          corner(1,5)=corner(1,1)
          corner(2,5)=corner(2,1)
          do j=1,4
            write (3,580) ((corner(k,1),k=1,2), i=j, j=1)
          enddo
        enddo
      endif
c
      write (3,580) ((frac(k,j),j=4,7),k=1,nfrac)
      close(unit=3)
c
      if (lplot.eq.-1) then
        open (unit=3, file=filer, status='unknown',
        , carriagecontrol='list')
        write (filer(6:7),20) 0
20 format (i2.2)
        if (lgene.eq.0) then
          write (3,300) title, xgene, ygene, xgene, ygene, 0.
          else
            write (3,310) title, rgene, rgene, 0., 0.
          endif
          close(unit=3)
        elseif (lplot.eq.0) then
          open (unit=3, file=filer, status='unknown',
          , carriagecontrol='list')
          if (lgene.eq.0) then
            write (3,300) title, xgene, ygene, xmesh, ymesh, rotan
            else

```

```

            write (3,310) title, rmesh, rhole, xhole, yhole
          endif
          close(unit=3)
        endif
      return
300 format (/a/a//1p,'xgene- - ',e10.3,'ygene- - ',e10.3/
      1 , 'xmesh- - ',e10.3,'ymesh- - ',e10.3/
      2 , 'rotan- - ',e10.3)
310 format (/a/a//1p,'rmesh- - ',e10.3,'rhole- - ',e10.3/
      , 'xhole- - ',e10.3,'yhole- - ',e10.3)
580 format (1p,2(2e10.3,10x))
      end

```

```

open (unit=3, file=riier, status='unknown',
      carriagecontrol='list')
if (lgene.eq.0) then
write (3,300) title,xgene,ygene,xmesh,ymesh,rotan
else

```

```

      subroutine rectxy (x,y,n,dseed,xs,ys,ltol,lpols,slcos,slsin)
c
c *** this subroutine generates random fracture centers
c
      dimension x(n),y(n)
      double precision dseed
c
      toler=10.**ltol
      xt=xs*toler
      yt=ys*toler
      x2=xs/2
      y2=ys/2
      if (lpols.ne.1) then
c
        do 290 i=1,n
          x(i)=float(int(xt*ggubfs(dseed)))/toler-x2
          y(i)=float(int(yt*ggubfs(dseed)))/toler-y2
290      continue
c
        else
c
c *** fracture centers randomly located on scanline passing
c *** through center of generation region
c
          xt=xt/2
          yt=yt/2
          do 300 i = 1,n
c
            ran = toler*(ggubfs(dseed)-.5)
            x(i)=float(int(ran*slcos + xt))/toler-x2
            y(i)=float(int(ran*slsin + yt))/toler-y2
300      continue
          endif
c
          return
        end

```

```

subroutine rlimit (icont,maxfr,frac)
parameter (nc=10)
common/param/ itole,iranf,iunits,ikeep,imesh,iprnt,nfrac,
               iplot,igene
common/inew/   inew(100)
common/old/    old(100)
common/kut/    kutone,kut(100)
common/mask/   mask(2,100)
integer*2 mask
common/mesh/   xgene,ygene,rgene,xmesh,ymesh,rotan,
               rmesh,ahole,yhole,rhole,nsqene
common/next/   next(100)
common/pi/     pi180
common/set1/   nsets,iset1(20,0),rset1(20,11)

*****
c
c this subroutine truncates a fracture if one or both of its
c endpoints fall outside of the block of dimension xmesh by ymesh,
c and discards the fracture if it is outside the block.
c
c if the fracture is truncated, the coordinate of the endpoints
c and the length of the fracture are recalculated.
c
c nfrac is the number of fractures.
c
c one components of frac may be recalculated in this subroutine -
c   frac(1,2) = length,
c   frac(1,4) = x1, x coordinate of endpoint 1,
c   frac(1,5) = y1, y coordinate of endpoint 1,
c   frac(1,6) = x1, x coordinate of endpoint 2,
c   frac(1,7) = y1, y coordinate of endpoint 2.
c
c the components of frac used in this subroutine are ---
c   frac(1, 8) = a = -sin of orientation,
c   frac(1, 9) = b =  cos of orientation,
c   frac(1,10) = c,
c where a, b, and c are the coefficients in the equation of the
c line, ax + by + c = 0, on which fracture i lies.
c
c   kut(i) = truncation code.
c   next(i) = where the next truncated fracture sits
c
dimension frac(maxfr,10)
dimension t(4),ls(4)
c
k=1
n2=0
inext=0
r=rotan*pi180
cosr=cos(r)
sinr=sin(r)
xm2=xmesh/2
ym2=ymesh/2
c
do 170 i=1,nsets
c
m=0
c
n1=n2+1
n2=n1+iset1(i,2)-1
if (n2.lt.n1) go to 170
c

```

```

do 160 i=n1,n2
c
c calculate array t of intersection values
c
nt=0
next(1)=0
flen=frac(i,2)
x=frac(i,4)
y=frac(i,5)
denomx=cosr*frac(i,8)+sinr*frac(i,9)
denomy=sinr*frac(i,8)-cosr*frac(i,9)
prtx=cosr*y-sinr*x
prty=cosr*x+sinr*y
if (denomx.ne.0.) then
  ls(1)=1
  ls(2)=3
  t(1)=(prtx+ym2)/denomx
  t(2)=(prtx-ym2)/denomx
  nt=2
else
  if (abs(prtx).ge.xm2) go to 160
endif
if (denomy.ne.0.) then
  ls(nt+1)=2
  ls(nt+2)=4
  t(nt+1)=(prty+ym2)/denomy
  t(nt+2)=(prty-xm2)/denomy
  nt=nt+2
else
  if (abs(prty).ge.ym2) go to 160
endif
c
c sort array t of intersection values
c
j1=2
j2=nt
do while (j2.ge.j1)
  j2=j2-1
  do 20 j=j1,j2
    if (t(j-1).le.t(j)) go to 20
    tt=t(j-1)
    t(j-1)=t(j)
    t(j)=tt
    it=ls(j-1)
    ls(j-1)=ls(j)
    ls(j)=it
    j2=j-1
  20 continue
  j1=max0(j1,2)
enddo
j1=nt/2
if (mod(ls(j1)+ls(j2)+1,2).eq.0) go to 160
if (t(j1).ge.flen.or.t(j1+1).le.0.) go to 160
c
c truncate fracture to block
c
kut(i)=0
if (t(j1+1).le.flen) then
  kut(i)=ls(j1+1)

```

```

n2=n1+1setf(1,2)-1
if (n2.lt.n1) go to 170

```

```

kut(1)=0
if (t(11+1).le.flen) then
kut(1)=15(11+1)

```

```

      frac(1,2)=t(11+1)
      frac(1,6)=x*frac(1,9)*t(11+1)
      frac(1,7)=y-frac(1,8)*t(11+1)
    endif
    if (t(11).ge.0.) then
      kut(1)=kut(1)+16*is(11)
      frac(1,2)=frac(1,2)-t(11)
      frac(1,4)=x*frac(1,9)*t(11)
      frac(1,5)=y-frac(1,8)*t(11)
    endif
c *** store information about fractures part or all of which fall into
c *** the flow region.
c
      ix1=(cosr*frac(1,4)+sinr*frac(1,5)*xm2)*nc/xmesh
      ix2=(cosr*frac(1,6)+sinr*frac(1,7)*xm2)*nc/xmesh
      iy1=(-sinr*frac(1,4)+cosr*frac(1,5)*ym2)*nc/ymesh
      iy2=(-sinr*frac(1,6)+cosr*frac(1,7)*ym2)*nc/ymesh
      ix=min(ix1,ix2)
      iy=min(iy1,iy2)
      ix2=ix1+ix2-ix+1
      iy2=iy1+iy2-iy+1
      ix2=2**min(ix2,nc)
      iy2=2**min(iy2,nc)
      ix=max(ix,0)
      iy=max(iy,0)
      ix1=2**min(nc-1,ix)
      iy1=2**min(nc-1,iy)
      mask(1,k)=ix2-ix1
      mask(2,k)=iy2-iy1
      kut(k)=kut(1)
      do 150 j=1,10
        frac(k,j)=frac(1,j)
150
c
      next(k)=0
      if (icont.lt.5) then
        if (kut(k).ne.0) then
          next(k)=inext
          inext=k
        endif
      elseif (icont.eq.5) then
        k16=kut(k)/16
        if (k16.eq.1.or.kut(k)-16*k16.eq.1) then
          next(k)=inext
          inext=k
        endif
      endif
      k=k+1
      m=m+1
c
160 continue
c
      iset1(1,2)=m
c
170 continue
c
      nfrac=k-1
      if (icont.lt.5) then
        if (kut(nfrac).eq.0) next(nfrac)=inext
      elseif (icont.eq.5) then
        k16=kut(nfrac)/16
        if (k16.ne.1.and.kut(nfrac)-16*k16.ne.1) next(nfrac)=inext

```

```

return
end

```

170 continue

```

nfrac=k-1
if (icont.lt.5) then
  if (kut(nfrac).eq.0) next(nfrac)=inext
elseif (icont.eq.5) then
  k16=kut(nfrac)/16
  if (k16.ne.1.and.kut(nfrac)-16*k16.ne.1) next(nfrac)=inext

```

```

subroutine spatia (maxfr, dseed, frac, i, n1, n2, idens)
common/param/ itole, iranf, iunits, ikeep, imesh, iprnt, nfrac,
common/mesh/ xgene, ygene, rgene, xmesh, ymesh, rotan,
common/set1/ nset, iset1(20,9), rset1(20,11)

```

this subroutine is called by fragen when generation is done by subregions.

```

icent = 1 - read in all fracture characteristics,
2 - generate fracture center coordinates,
3 - generate fracture center coordinates by subregions

```

```

idens = 0 - read and use r1amb
1 - read nfrac directly
2 - calc. r1amb from lbar & r1ambdal

```

```

ipols = 0 - generate fracture centers in whole gen. region
ichar - read value for each of orientation, length and aperture,
2 - set characteristic equal to a constant value,
3 - generate characteristics values according to a distribution, (local if icent=3)
4 - generate apertures correlated to log of length
5 - generate apertures correlated to length
6 - generate subregion data using global parameters from tapel
7 - generate a mean value using the global parameters, read a local standard deviation to generate local values.

```

```

idist = 1 - normal distribution,
2 - lognormal distribution,
3 - exponential distribution,
4 - gamma distribution.

```

In addition, the following variables are read in ---

```

nfrac = number of fractures,
r1ambn = no. of fractures per square subunit,
itole = number of decimal places in fracture center coordinates
const = see ichar = 2,
ev = expected value of statistical distribution, or lambdal for gamma distr.
sd = standard deviation of statistical distribution, or alphas for gamma distr.

```

the array iset1 is used to store the values of nfrac, icent, and ichar and idist for each set. the array rset1 is used to store the global values of const, ev and sd for each set.

```

dimension frac(maxfr,10)
real nev(1,1)

```

```
double precision dseed, rseed(3), mseed(3)
```

Initialise seeds

dseed is used to generate fracture centers.
mseed is used to generate means for local distributions.
rseed is used to generate orientation, length, or aperture.

```

xg2=xgene/2
yg2=ygene/2
rseed(1)=dseed
mseed(1)=dseed
do 170 ll=2,3
  rseed(ll)=rseed(ll-1)-1.d0
  mseed(ll)=rseed(ll)

```

```

170 continue
read global data on tapel
read(1,250) rset1(1,11)

```

```

l=1
m=2
do 180 k=1,3
  l=l+2
  m=m+3

```

```

read (1,240) ichar
iset1(1,1)=ichar

```

*** read in code for statistical distribution and read in
*** statistical parameters

```

if (ichar.eq.3.or.ichar.ge.6) then
  read (1,260) idist, ev, sd
  iset1(1,1)=idist
  if (idist.eq.3) sd=ev
  rset1(1,m+1)=ev
  rset1(1,m+2)=sd
elseif (ichar.gt.3) then
  read (1,270) ycept, slope, sd
  iset1(1,1)=1
  rset1(1,m)=ycept
  rset1(1,m+1)=slope
  rset1(1,m+2)=sd
endif

```

```
180 continue
```

*** generate locations of fracture centers
spatia fills the subregions moving to the right starting from top left corner, ie top row 1 to r, 2nd row 1 to r etc.

```

xsubgene=xgene/nsgene
ysubgene=ygene/nsgene
iset1(1,2)=0
rset1(1,10)=0.
do 230 nn=1,nsgene
  do 220 mm=1,nsgene
    if (idens.eq.0) then
      read (7,250) r1ambn
      nfracnn=nint(xgene*ygene*r1ambn/nsgene**2)
    elseif (idens.eq.2) then
      read(7,250) r1bar, r1ambdal

```



```

c      lchar and ldist for each set. the array rseti is used to store
c      the global values of const, ev and sd for each set.
c      .....
c      dimension frac(maxfrc,10)
c      real nev(1,1)
c

```

```

rseti(1,10)=0.
do 230 nn=1,nsgene
do 220 mm=1,nsgene
if (ldens.eq.0) then
read (7,250) rlammm
nfracmn=nint(xgene*ygene*rlammm/nsgene**2)
elseif (ldens.eq.2) then
read(7,250) r1bar, r1mbdal

```

```

c      if(r1bar.le.1.e-70)stop
c      'mean length in subregion should be strictly positive'
c      rlammm= r1mbdal / r1bar
c      nfracmn=nint(xgene*ygene*rlammm/nsgene**2)
c      else
c      read (7,240) nfracmn,theta
c      rlammm=nfracmn/(xgene*ygene*nsgene**2)
c      endif
c
c      if (nfracmn.eq.0)goto 220
c      if (lpolr.eq.1) stop 'lpolr cannot be 1 here'
c      n2=nfracmn+1
c      lseti(1,2)=nfracmn+lseti(1,2)
c      rseti(1,10)=rlammm/nsgene**2+rseti(1,10)
c
c      call rectxy (frac(n1,4),frac(n1,5),nfracmn,dseed,
c      xsubgene,ysubgene,ltole,lpolr,dumb1,dumb2)
c
c      adjust centers
c
c      do 190 kk=n1,n2
c      frac(kk,4)=frac(kk,4)+(xq2*(2*mm-nsgene-1))/nsgene
c      frac(kk,5)=frac(kk,5)+(yq2*(nsgene+1-2*nn))/nsgene
c      continue
c
c      190
c      i=1
c      m=2
c      do 210 k=1,3
c      i=i+2
c      m=m+3
c
c      lchar=lseti(1,1)
c      ldist=lseti(1,i+1)
c      ev=lseti(1,m+1)
c      sd=rseti(1,m+2)
c
c      if (lchar.eq.4.or.lchar.eq.5) then
c      read (7,270) ycept,slope,sd
c      call normdl (frac(n1,k),n2-n1+1,rseed(k),ycept,sd,slope,
c      frac(n1,2),lchar)
c      elseif (lchar.ge.3.and.lchar.le.7) then
c
c      *** read in code for statistical distribution and read in
c      *** statistical parameters
c
c      if (lchar.eq.3) then
c      read (7,270) ev,sd
c
c      otherwise keep global values
c
c      elseif (lchar.eq.7) then
c
c      call distrl (ldist,nev,1,mseed(k),ev,sd)
c      ev=nev(1,1)
c      read(7,270)sd
c      endif
c
c      *** patch to allow ev=lambda & sd=alpha
c
c      if (ldist.eq.4) then
c      ev=sd/ev
c      sd=sqrt(ev*ev/sd)
c      endif

```

```

c      call distrl (ldist,frac(n1,k),n2-n1+1,rseed(k),ev,sd)
c      else
c      *** set fracture characteristic to a constant
c
c      read (7,280) const
c      do 200 j=n1,n2
c      frac(j,k)=const
c      continue
c      200
c      endif
c
c      210
c      continue
c
c      n1=n2+1
c
c      220
c      continue
c
c      230
c      continue
c      return
c
c      240
c      format (10x,15,15x,f10.4)
c      250
c      format (10x,e10.3,10x,f10.4)
c      260
c      format (10x,15,15x,f10.4,10x,f10.4)
c      270
c      format (3(10x,f10.0))
c      280
c      format (10x,f10.4)
c      end

```

```

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

```

- 84 -

Pages 84-90 intentionally removed.

```

      subroutine wrenum(maxfrac,frac)
c
c   this subroutine writes renum22.dat
c   and uses the following arrays
c   frac - array of fracture information
c   kut - index into array knode
c   knode - index array into tint and ifrac
c   tint - t value of intersection
c   ifrac - fractures in intersection
c
c   common/files/ iplot,irenum
c   common/param/ itole,ixanf,iunits,ikeep,imesh,iprnt,nfrac,
c   iplot,igene
c   common/qc/ visc,spgr,qc,itrans
c   common/mesh/ xgene,ygene,rgene,xmesh,ymesh,rotan,
c   rmesh,xhole,yhole,rhole,nx,ny
c   common/title/ title,iray,irate
c   common/kut/ kutone,kut(1)
c   common/knode/ knode(1)
c   common/loid/ loid(1)
c   common/inew/ inew(1)
c   common/ifrac/ ifrac(2,1)
c   common/lines/ nnodes,nelene
c   common/kind/ kind(1)
c   byte kind
c   common/tint/ tint(2,1)
c   common/pi/ pi180
c
c   dimension frac(maxfrac,10)
c   dimension nintside(4)
c   character*80 title(2)
c   dimension bvalu(5),ia(2),a1(4),b1(4),a2(4),b2(4),c(5),bval(2)
c   character*9 idate
c   character*19 iray
c   character*11 file
c   integer bcode(5)
c
c   toler=10.**(-itole-1)
c   do i=1,4
c     nintside(i)=0
c   enddo
c
c   *** read in the types and values of the boundary conditions.
c
c   read (1,100) (bcode(i),i=1,4)
c   read (1,110) (bvalu(i),i=1,4)
c
c   if (nfrac.eq.0) write(6,120) rotan
c   irenum=irenum+1
c   write(file,125) irenum
c125 format('renum',i2.2,'.dat')
c   open (unit=4,file=file,status='unknown',
c     carriagecontrol='list')
c
c   *** print the contents of frac and kut.
c
c   write(6,130) iray,irate
c   write(6,140)
c   write(6,150)
c   write(6,160) xmesh,ymesh,rotan
c   write(6,170) nfrac
c
c   write(6,180)
c   write(6,190)

```

```

c
c   if (imesh.eq.0) then
c     write(6,200)
c     stop
c   endif
c
c   ***** f i n i t e   e l e m e n t   m e s h *****
c
c   *** write input deck for finite element program.
c
c   *** set coeff. for calculating linear potential boundary
c   *** conditions.
c
c   if (igene.eq.0) then
c     bcode(5)=bcode(1)
c     bvalu(5)=bvalu(1)
c
c     r=rotan*pi180
c     cosr=cos(r)
c     sinr=sin(r)
c     a1(1)=(bvalu(4)-bvalu(2))*cosr
c     b1(1)=(bvalu(4)-bvalu(2))*sinr
c     a2(2)=b1(1)/ymesh
c     b2(2)=a1(1)/ymesh
c     a1(1)=a1(1)/xmesh
c     b1(1)=b1(1)/xmesh
c     c(1)=(bvalu(2)+bvalu(4))/2
c     a1(2)=(bvalu(3)-bvalu(1))*sinr
c     b1(2)=(bvalu(3)-bvalu(1))*cosr
c     a2(1)=b1(2)/xmesh
c     b2(1)=a1(2)/xmesh
c     a1(2)=a1(2)/ymesh
c     b1(2)=b1(2)/ymesh
c     c(2)=(bvalu(3)+bvalu(1))/2
c     do i=1,2
c       a1(i+2)=a1(i)
c       b1(i+2)=b1(i)
c       a2(i+2)=a2(i)
c       b2(i+2)=b2(i)
c       c(i+2)=c(i)
c     enddo
c     c(5)=c(1)
c
c   *** write out arrays on tape4 in format for finite element program.
c
c   maxd=2
c   ibkey1=bcode(4)
c   if (ibkey1.eq.2) ibkey1=1
c   do i=1,4
c     ibkey2=bcode(i)
c     if (ibkey2.eq.2) ibkey2=1
c     if (ibkey1.ne.ibkey2) maxd=1
c     ibkey1=ibkey2
c   enddo
c   else
c     maxd=1
c   endif
c   write (4,210) iray,irate,iprnt
c   write (4,220) title
c   write (4,230) nfrac
c   if (igene.eq.0) then
c     write (4,240) xgene,ygene
c     write (4,250) xmesh,ymesh,rotan
c   else

```

```

c
write(6,180)
write(6,190)

```

```

write(4,240) xgene,ygene
write(4,250) xmesh,ymesh,rotan
else

```

```

write(4,130) rgate
write(4,140) rmesh,rhole,xhole,yhole
endif
write(4,260) (bcode(i),i=1,4)
write(4,270) (bval(i),i=1,4)
write(4,280) visc,spqr,nelcmm,nnodes,maxd,ikeep,iplot
c
write(6,290) rotan,nelcmm,nnodes
c
write nodes
c
inode=0
int3=0
ifrac=nfrac
do i=1,nfrac
  io=iold(i)
  told=-2*toler
  int1=int3+1
  int2=kut(i)
  int3=int2
  do while (knode(int2).eq.0.and.int2.ge.int1)
    int2=int2-1
  enddo
  k=knode(int1)
  if (int2.lt.int1) then
    k=0
  else
    k2=(1-frac(1,k))/(frac(2,k)-frac(1,k))
    if ((k2.and..not.1).ne.0) stop ' k2 error 6'
    t=tint(k2+1,k)
  endif
  if (ikeep.ne.0.and.(k.eq.0.or.t.gt.toler)) then
    inode=inode+1
    ibkey1=0
    iside=0
    told=0.
    x=frac(io,4)
    y=frac(io,5)
    write(4,300) inode,ibkey1,side,x,y
  endif
  if (k.ne.0) then
    do k1=int1,int2
      k=knode(k1)
      k2=(1-frac(1,k))/(frac(2,k)-frac(1,k))
      if ((k2.and..not.1).ne.0) stop ' k2 error 7'
      t=tint(k2+1,k)
      ifrac=frac(2-k2,k)
      if (t.gt.told*toler) then
        x=frac(io,4)+frac(io,9)*t
        y=frac(io,5)-frac(io,8)*t
      endif
      told=t
    enddo
    if (ifrac.lt.0) then
      inode=inode+1
      iside=-ifrac
      nintside(iside)=nintside(iside)+1
      ibkey1 = bcode(iside)
      bval(1) = bval(iside)
      if (lgene.eq.1.and.iside.eq.1.and.ibkey1.eq.-1) then
        iside=7
        x=xhole
        y=yhole
      else
        ibkey2 = bcode(iside+1)

```

```

      bval(2) = bval(iside+1)
    endif
  c
  ccc linear potential on sides
  c
  if (ibkey1.eq.2) then
    bval(1)=a1(iside)*x+b1(iside)*y+c(iside)
    ibkey1 = 1
  endif
  c
  if (ibkey2.eq.2) then
    bval(2)=a2(iside)*x+b2(iside)*y+c(iside+1)
  endif
  write(4,300) inode,ibkey1,side,x,y,
    (bval(1b),1b=1,maxd)
  kind(k)--2
  elseif (kind(k).ne.-2) then
    inode=inode+1
    ibkey1=0
    iside=0
    write(4,300) inode,ibkey1,side,x,y
    kind(k)--2
  endif
enddo
endif
if (ikeep.ne.0.and.(k.eq.0.or.told*toler.lt.frac(io,2))) then
  inode=inode+1
  ibkey1=0
  iside=0
  write(4,300) inode,ibkey1,side,frac(io,6),frac(io,7)
endif
enddo
c
write elements
c
inode=0
int3=0
ieleme=0
do i=1,nfrac
  io=iold(i)
  trans = frac(io,3)
  if (itrans.eq.0) trans=qc*trans**3
  int1=int3+1
  int2=kut(i)
  int3=int2
  do while (knode(int2).eq.0.and.int1.le.int2)
    int2=int2-1
  enddo
  int2=0
  told=0.
  k=knode(int1)
  if (int2.lt.int1) then
    k=0
  else
    k2=(1-frac(1,k))/(frac(2,k)-frac(1,k))
    if ((k2.and..not.1).ne.0) stop ' k2 error 9'
    t=tint(k2+1,k)
    r1=t
    t2=t
  endif
  if (ikeep.ne.0.and.(k.eq.0.or.t.gt.toler)) then
    inode=inode+1
    int2=inode
  endif

```

```

if (lkeep.ne.0.and.(k.eq.0.or.t.gt.toler)) then
  inode=inode+1
  in2=inode
endif
endif

```

```

c      if (f1.eq.f2) then
c          frac(10,1)=ygene
c          frac(10,6)=xgene
c          frac(10,5)=ygene
c          frac(10,4)=xgene
c          endif
c      *
c      endif
c      enddo
c
c      write the number of boundary nodes per side on fmq.out
c
c      write(6,6001) nints1side
c      close (unit=4)
c      return
c
100 format(4110)
110 format(4f10.4)
120 format('Other areas are no conducting fractures for rotan=',f6.2)
130 format('0',a19,'-',a9)
140 format('Of fractures in flow region')
150 format('Of non-intersecting fractures',
1 ' have been dropped')
160 format('The size of the flow region is ',f8.1,' by ',f8.1/
1 ' the angle of rotation is ',f6.2)
170 format('The number of fractures in the flow region is ',i5)
180 format('Of fracture statistics')
190 format('Of fractures in flow region')
190 format('Isolated fractures have been eliminated')
200 format('Onormal end of generation,lnesh=0')
210 format(a19,'-',a9,i1)
220 format(a)
230 format('nfrac--',i5)
240 format('xgene--',f10.4,'ygene--',f10.4)
250 format('xmesh--',f10.4,'ymesh--',f10.4,'rotan--',f10.4)
260 format('bcode(s)--',4110)
270 format('bvalu(s)--',4f10.4)
280 format('vlocality--',f10.4,'apgr--',f10.4/
1 'nlements--',i6,5x,'nnodes--',i6,5x,'nboundary--',i6,5x/
2 'trunc.code--',i5,5x,'plot code--',i5)
290 format('rotan--',f10.4,5x,'nelem--',i6,5x,'nnode--',i6)
300 format(3i5,5x,2f10.4,10x,2f10.4)
320 format(3i5,5x,1p,2e10.3,5x,i5)
330 format('rgene--',f10.4)
340 format('xmesh--',f10.4,'xhole--',f10.4/
'xhole--',f10.4,'yhole--',f10.4)
6001 format('Number of boundary intersections per side:')
' side one ',i10/
' side two ',i10/
' side three ',i10/
' side four ',i10)
end

```

Appendix C

RENUM - Program Organization and Arrays

Table C-1. RENUM - Description of Program Variables

Variable	Description	How Value is Assigned
Global:		
ikeep	Code for discarding dead-ends = 0 - keep only conducting fracture network = 1 - keep dead-ends = 2 - keep dead-ends and isolated clusters	Read in RDATA
iplot	Code for producing input files for the plotting program DIMES = 0 no plot = 1 - generate plot files after RENUM only = 2 - generate plot files after both FMG and RENUM	Read in RDATA
maxd	Number of boundary conditions	Read in RDATA
mxelem	Maximum number of elements, used for dimensioning arrays	Set in parameter statement in main program
mxnode	Maximum number of nodes, used for dimensioning arrays	Set in parameter statement in main program
neleme	Number of elements	Read in RDATA
nflow	Number of current flow region	Initialized to zero incremented in RDATA
newnel	Number of elements after mesh optimization	Calculated in MCGEE
nnodes	Number of nodes	Read in RDATA, modified in PATHS
Local:		
ibwth	Bandwidth of the linear system after mesh optimization	Computed in PROUT
ie	Current element number	Indexed in MCGEE, MERGE PATHS, PLOT, PROUT
ihole	Number of parallelepiped with constant flux boundary conditions, or "hole"	Computed and used in MERGE

ilasd	Last node added in the current list of nodes for the downward search	Set and incremented in PATHS
ilasf	Last node added in the current list of nodes for the forward search	Set, incremented in PATHS
in	Current node new number	Indexed in MCGEE, MERGE, PATHS, PLOT and PROUT
ind	Number of nodes already connected	Calculated in PATHS
ind1	First node of the current level	Determined in PATHS
ind2	Last node of current level	Determined in PATHS
io	Old number of the current node, in	Set in MCGEE, MERGE, PATHS, PLOT and PROUT
is	Source number of the current node, in	Determined and used in PATHS
isour	Flag for marking a node as a source for the next forward search, used during downward searches	Set in PATHS
jn	New number of node connected to in by element ie	Used McGee, PATHS, PLOT and PROUT
jo	Old number for node jn	Set in MCGEE, PATHS, PLOT and PROUT
js	Source number of the node jn connected to the current node in	Determined and used in PATHS
knode	Number of nodes already connected	Initialized and incremented in MCGEE
knode1	First node of the current level	Determined in MCGEE
knode2	Last node of the current level	Determined in MCGEE
next0	Dummy variable used to set up the "next" common property	Used in PATHS
nodeso	Initial number of nodes	Set in PATHS

Table C-2. RENUM - Description of Program Arrays

Array	Description
aptdis(2,mxelem)	Element characteristics; read in subroutine RDATA; m = 1, neleme aptdis(1,m) = transmissivity aptdis(2,m) = element length
ibeg(10)	Number of the first boundary node encountered on a given imposed flux parallel piped or circle ("hole") ihole = 1, 10 ibeg(ihole) = number of first node on hole number ihole. All subsequent nodes on hole ihole will be shrunk into node ibeg(ihole) Defined and used in MERGE
ibs(2,mxnode)	Node information; n = 1, nnodes ibs(1,n) = node type = 0 internal node = 1 imposed head boundary node = -1 imposed flux boundary node ibs(2,n) = boundary side number for boundary nodes
ielte(2,mxelem)	element connections array, ie = 1, neleme; i = 1, 2 ielte(i, ie) = number of the next element connected to element ie through its endpoint number i
ieltn(mxnode)	Element connection pointer, n = 1, nnodes ieltn(n) = number of first element in the list of elements connected to node n
ifr(2,mxelem)	fracture numbers, ie = 1, neleme ifr(1,ie) = number of the fracture on which lies an element (fracture line in 2-D; fracture disc in 3-D) ifr(2,ie) = number of other fracture disc on which lies an element, if relevant (3-D only)
iold(mxnode)	Reference array for old node numbers, in = 1, nnodes iold(in) = old number of node number in

inew(mxnode)	Back reference in array for new node numbers, io = 1, nnode iold(io) = 0 if node with old number io is discarded iold(io) = new node number otherwise
inode(2,mxelem)	Node numbers of the two nodes defining an element ie = 1, neleme inode(1,ie) = number of the first node of element ie inode(2,ie) = number of the second node of element ie
isrc(mxnode)	source number array, in = 1, nnodes isrc(in) = source of node in
next (mxnode)	Pointer array used for searches in PATH n = 1, nnodes next(n) = node to be screened after node n in the current search.
title(10)	First ten lines read from RENUM%%.inp then written to linel.inp: information needed by LINEL but not by RENUM. (character*80)
xyzphi(6,mxnode)	Node information, n = 1, nnodes xyzphi(1,n) = x-coordinate of node n xyzphi(2,n) = y-coordinate of node n xyzphi(3,n) = z-coordinate of node n xyzphi(4,n) = value of the imposed head at node n, first boundary conditions xyzphi(5,n) = value of the imposed head at node n, second boundary conditions xyzphi(6,n) = value of the imposed head at node n, third boundary conditions

Table C-3. RENUM - Subroutine Outline

RENUM	RDATA	WERROR
MERGE		
PATHS		
MCGEE		
PLOT		
PROUT		
or		
RCNTRL		
RNPN		
TRIOUT		

Table C-4. RENUM - Description of Subroutines

Subroutine	Description
MCGEE	<p>Sorts nodes by coordinates and then renumbers them using the Cutill-McKee method.</p> <p>given: <code>inew(nnodes), iold(nnodes), ielte(2,neleme), ieltn(nnodes), ibs(2,nnodes), nnodes, ikeep, xyzphi(6,nnodes), inode(2, neleme)</code> returns: <code>rewnel, modified nnodes, inew, iold</code></p>
MERGE	<p>Eliminates zero-length elements, and merges no flow boundary nodes</p> <p>given: <code>inew(nnodes), iold(nnodes), ieltn(nnodes), ibs(2,nnodes), nnodes, neleme, xyzphi(6,nnodes), inode(2,neleme), aptdis(2,neleme), ifr(2,neleme)</code> returns: <code>ielte, modified ibs, inew, iold, inode, aptdis, ifr, neleme, ieltn</code></p>
PATHS	<p>Traces connected paths from the boundaries in order to detect and discard complex dead-ends</p> <p>given: <code>inew(nnodes), iold(nnodes), nnodes, ibs(2,nnodes), ielte(2,neleme), ieltn(nnodes), inode(2,neleme)</code> returns: <code>modified inew, iold, nnodes</code></p>
PLOT	<p>Writes plotting files, lines%.dat, which are used by program DIMES to produce fracture plots</p> <p>given: <code>nnodes, ielte(2,neleme), ieltn(nnodes), inode(2, neleme), ifr(2,neleme), xyzphi(6,nnodes), inew(nnodes), iold(nnodes)</code> prints: endpoints coordinates and fracture(s) number(s) for each element in the optimized network.</p>
PROUT	<p>Prints program output to the file line1.inp.</p> <p>given: <code>title, newnel, nnodes, maxd, ikeep, iplot, ibwth, inew(nnodes), iold(nnodes), ibs(2, nnodes), xyzphi(6, nnodes), ielte(2, neleme), ieltn (nnodes), inode (2, neleme), ifr (2, neleme), aptdis(2, neleme)</code> prints: - a header containing title, newnel, nnodes, maxd, ikeep iplot, ibwth - a list of nodes with a node number, and the information in ibs and xyzphi</p>

- a list of elements with element number, numbers of the two nodes making up the element, and the information contained in aptdis and ifr

RCNTRL Reads control variables from inter.inp

returns: title2, imode, ibmode, mxstep, iue, iui,
nstep, time0, dtini, prr, tmax, theta,
fsys, tole, dcint, dcon, dcoff, rhor, rmur, gravr,
ssubs, disp

RDATA Reads in data from renum%%.inp and also checks the problem size against the array dimensions.

given: nflow
reads: all input (see Table 3-1)
returns: all it reads, and incremented nflow
uses subroutine: WERROR

TRIOUT Prints program output to files ctrl.inp, node.inp and elmt.inp

given: type in the "givens" for PROUT and the "returns" for RCNTRL

prints: - a control file ctrl.inp for input to TRINET to node.inp
- a list of nodes with a node number, and the information in ibs and xyzphi
- to elmt.inp a list of elements with element number members of the two nodes making up the element, and the information contained in aptdis and ifr

WERROR Prints error messages in sys\$error file.

given: character*(*) line
prints: error message contained in line

Appendix D

RENUM - Program Listing

```

program renum
parameter (mxnode=50000)
parameter (mxelem=90000)
common/aptdis/ aptdis(2,mxelem)
common/control/mxele,mxnod
common/forward/nnodes,nelems,maxd,newnel,nflow,ikeep,iplot
common/lbs/ lbs(2,mxnode)
common/ielte/ ielte(2,mxelem)
common/ieltl/ ieltl(mxnode)
common/lfr/ lfr(2,mxelem)
common/lfrac/ lfrac(mxnode)
common/lnew/ lnw(mxnode)
common/inode/ inode(2,mxelem)
common/iold/ iold(mxnode)
common/lsrc/ lsrc(mxnode)
common/next/ next(0:mxnode)
common/title/ title
common/xyzphi/ xyzphi(6,mxnode)
common /cgeo/ dc(mxelem),ss(mxelem),w(mxelem)
common /contrl/ title2,imode,ihmode,mxstep,iwo,iul,
$ nstep,time0,dtini,prf,tmax,theta,fsys,
$ tole,dcint,dcon,dcoff,rhor,rnur,gravr,
$ ssubs,dispc
character*80 title(10),title2
logical*1 fxl
mxnod=mxnode
mxele=mxelem
open (unit=4,file='line1.inp',status='unknown')
nflow=0
do while (.true.)
  call rdata
c
  do i=1,nnodes
    ieltl(i)=0
    iold(i)=1
    lnw(i)=1
  enddo
c
c  Merge inner constant
c  flux boundary nodes and get rid of zero length elements
c
  call merge
c
c  trace connected paths
c
  if(ikeep.eq.0)call paths
c
c  renumber the nodes using CutHill-McKee's method
c
  call mcgee
c
c  write a file for plotting
c
  if(iplot.ge.1)call plot
c
c  write line1 input deck
c
  inquire(file='inter.inp',exist=fxl)
  if(.not.fxl) then
    call prout
  else
    call rcntrl
    call rnpr
    call triout

```



```
    end if  
  enddo  
end
```

```

parameter (maxno=18000,maxel=34000,maxbn=4000)
common /imgb/      ibcode(6),bvalufmg(6)
common /bound/     lb(maxno),lbc(maxno),bvalu(maxno),cbvalu(maxno),
$                  isc(maxno),nboun,kb(6)
common /cgeo/      dc(maxel)
common /cont/      cold(maxno)
common /ctrl/      title1,title2,imode,ibmode,mxstep,iuo,iui,
$                  nstep,time0,dtini,prf,tmax,theta,fsys,
$                  tole,dcint,dcon,dcoff
common /qed/       ne,nh,ibw,icat(2,maxel),xyz(3,maxno),
$                  rl(maxel),w(maxel),ss(maxel),transm(maxel),
$                  ssubs,dispc
common /head/      h(maxno)
common /para/      rhox,rmur,gravr
character          title1(3)*80,title2*80
logical*1          finit,fal

```

```

subroutine mcgee
common/forward/nnodes,neleme,maxd,newnel,nflow,ikeep,iplot
common/xyzphi/xyzphi(6,1)
common/ieltn/ieltn(1)
common/iold/iold(1)
common/inew/inew(1)
common/ielte/ielte(2,1)
common/lbs/lbs(2,1)
common/inode/inode(2,1)

```

```

c
knode2=0
knode=0
j=0
do while (knode.lt.nnodes.and.j.le.500)
  j=j+1
  j9=500
  knode3=knode+2
  knp11=knode+1
  do ln=knp11,nnodes
    io=iold(ln)
    if (lbs(2,io).ne.0.and.ieltn(io).ne.0) then
      if (lbs(2,io).eq.j) then
        knode=knode+1
        ko=iold(knode)
        iold(knode)=io
        inew(io)=knode
        iold(ln)=ko
        inew(ko)=ln
      else
        j9=min(j9,lbs(2,io)-1)
      endif
    endif
  enddo
  if (ikeep.gt.1.and.j.eq.501) then
    j=500
    do while (nnodes.gt.knode.and.knode.lt.knp11)
      io=iold(knp11)
      if (ieltn(io).eq.0) then
        ko=iold(nnodes)
        iold(nnodes)=io
        inew(io)=nnodes
        iold(knp11)=ko
        inew(ko)=knp11
        nnodes=nnodes-1
      else
        knode=knp11
      endif
    enddo
  endif

```

```

c
c
c
sort array by coordinates

```

```

  j1=knode3
  i2=knode
  do while (i2.ge.j1)
    j2=i2
    i0=i1
    i1=0
    i2=0
    in1=iold(j1-1)
    do j0=j1,j2
      in2=iold(j0)
      xt=xyzphi(1,in1)
      yt=xyzphi(2,in1)

```

```

        zt=xyzphi(3,in1)
        if (xt.gt.xyzphi(1,in2).or.
2         (xt.eq.xyzphi(1,in2).and.(yt.gt.xyzphi(2,in2).or.
3         (yt.eq.xyzphi(2,in2).and.zt.gt.xyzphi(3,in2)))) then
            i2=j0-1
            iold(j0)=in1
            iold(i2)=in2
            inew(in1)=j0
            inew(in2)=i2
            i1=i1+10*i2
            i0=0
        else
            in1=in2
        endif
    enddo
    j1=max0(i1,knode3)
enddo

```

```

do while (knode.gt.knode2)
    knode1=knode2+1
    knode2=knode

```

```

do in=knode1,knode2
    knode3=knode+2
    io=iold(in)
    ie=ieltm(io)
    do while (ie.ne.0)
        k=(io-inode(1,ie))/(inode(2,ie)-inode(1,ie))
        jo=inode(2-k,ie)
        jn=inew(jo)
        if (jn.gt.knode.and.jn.le.nnodes) then
            knode=knode+1
            ko=iold(knode)
            iold(knode)=jo
            inew(jo)=knode
            iold(jn)=ko
            inew(ko)=jn
        endif
        ie=ielte(k+1,ie)
    enddo
enddo

```

```

c
c  sort array by coordinates
c

```

```

    j1=knode3
    i2=knode
    do while (i2.ge.j1)
        j2=i2
        i0=1
        i1=0
        i2=0
        in1=iold(j1-1)
        do j0=j1, j2
            in2=iold(j0)
            xt=xyzphi(1,in1)
            yt=xyzphi(2,in1)
            zt=xyzphi(3,in1)
            if (xt.gt.xyzphi(1,in2).or.
2             (xt.eq.xyzphi(1,in2).and.(yt.gt.xyzphi(2,in2).or.
3             (yt.eq.xyzphi(2,in2).and.zt.gt.xyzphi(3,in2)))) then
                i2=j0-1
                iold(j0)=in1
                iold(i2)=in2
                inew(in1)=j0
                inew(in2)=i2
            endif
        enddo
    enddo

```

```

        i1=i1+10*i2
        i0=0
    else
        i0=i1
    endif
enddo
j1=max0(i1,knode3)
enddo
enddo
enddo
j=max(j,j9)
enddo

nnodes=knode
newnel=0
do i=1,nelms
    n1=inode(1,i)
    n2=inode(2,i)
    n1=lnw(n1)
    n2=lnw(n2)
    if (n1*n2.ne.0) newnel=newnel+1
enddo
return
end

```

```

subroutine merge
common/title/title
common/forward/nnodes,neleme,maxd,newnei,nflow,keep,ipiot
common/control/maxele,maxnod
common/ieltn/ieltn(1)
common/iold/iold(1)
common/inew/inew(1)
common/ielte/ielte(2,1)
common/xyzphi/xyzphi(6,1)
common/lbs/lbs(2,1)
common/inode/inode(2,1)
common/aptdis/aptdis(2,1)
common/lfr/lfr(2,1)
character*80 title(10)
integer lbeg(20)

```

```

merge inner constant flux boundary elements

```

```

initialize flag

```

```

do i=1,20
  lbeg(i)=0
enddo

```

```

loop over nodes

```

```

do in=1,nnodes
  if (lbs(1,in).eq.-1) then
    ihole=(lbs(2,in)-1)/6
    if (ihole.ne.0) then
      if (lbeg(ihole).eq.0) then
        lbeg(ihole)=in
      else
        lnew(in)=lbeg(ihole)
        lbs(2,in)=0
      endif
    endif
  endif
enddo

```

```

get rid of zero length elements

```

```

i=0
do while (i.lt.neleme)
  i=i+1
  n1=inode(1,i)
  n2=inode(2,i)
  do while (lnew(n1).ne.n1)
    n1=lnew(n1)
  enddo
  do while (lnew(n2).ne.n2)
    n2=lnew(n2)
  enddo
  if (aptdis(2,i).eq.0..or.n1.eq.n2) then
    if (lbs(2,n2).eq.0) then
      lnew(n2)=n1
    elseif (n1.ne.n2) then
      lnew(n1)=n2
      lbs(2,n1)=0
    endif
    inode(1,i)=inode(1,neleme)
    inode(2,i)=inode(2,neleme)
    aptdis(1,i)=aptdis(1,neleme)
    aptdis(2,i)=aptdis(2,neleme)
  endif
enddo

```

```

      ifr(1,1)=ifr(1,neleme)
      ifr(2,1)=ifr(2,neleme)
      i=i+1
      neleme=neleme-1
    endif
  enddo
  i=1
  do while (i.le.neleme)

```

```

c      get read of element if it is a loop on one node

```

```

      n1=inode(1,i)
      do while(inew(n1).ne.n1)
        n1=inew(n1)
      enddo
      inode(1,i)=n1
      n2=inode(2,i)
      do while (inew(n2).ne.n2)
        n2=inew(n2)
      enddo
      inode(2,i)=n2
      if(n1.eq.n2)then
        inode(1,i)=inode(1,neleme)
        inode(2,i)=inode(2,neleme)
        aptdis(1,i)=aptdis(1,neleme)
        aptdis(2,i)=aptdis(2,neleme)
        ifr(1,i)=ifr(1,neleme)
        ifr(2,i)=ifr(2,neleme)
        neleme=neleme-1
        goto 20
      endif

```

```

c      get rid of elements in parallel

```

```

      ie=ieltm(n2)
      do while (ie.ne.0)
        k=(n2-inode(1,ie))/(inode(2,ie)-inode(1,ie))
        ni=inode(2-k,ie)
        if (ni.eq.inode(1,i)) then
          if(aptdis(2,i).ne.0)
            aptdis(1,ie)=aptdis(1,ie)+aptdis(1,i)*
            aptdis(2,ie)/aptdis(2,i)
          inode(1,i)=inode(1,neleme)
          inode(2,i)=inode(2,neleme)
          aptdis(1,i)=aptdis(1,neleme)
          aptdis(2,i)=aptdis(2,neleme)
          ifr(1,i)=ifr(1,neleme)
          ifr(2,i)=ifr(2,neleme)
          neleme=neleme-1
          go to 20
        endif
        ie=ielte(k+1,ie)
      enddo
      do j=1,2
        n2=inode(j,i)
        ielte(j,i)=ieltm(n2)
        ieltm(n2)=i
      enddo
      i=i+1
20  enddo

```

```

c      return
      end

```

```

subroutine paths
common/forward/nnodes,neleme,maxd,newnel,nflow,ikeep,iplot
common/control/maxela,maxnod
common/ieltn/ieltn(1)
common/ielte/ielte(2,1)
common/iold/iold(1)
common/inew/inew(1)
common/isrc/isrc(1)
common/ibs/ibs(2,1)
common/inode/inode(2,1)
common/next/next0,next(1)
nodeso=nnodes

```

```

c
c initialize search. Put boundary nodes as sources for 1st search
c

```

```

      ilasf=0
      do i=1,nnodes
        if (ibs(2,i).ne.0) then
          next(ilasf)=i
          ilasf=i
        endif
        isrc(i)=i
      enddo
      next(ilasf)=0

```

```

c
c begin loop over searches
c

```

```

      i = next(0)
      do while (i.ne.0)

```

```

c
c         forward search
c

```

```

c         put sources for search in level one
c

```

```

      ind=0
      do while(i.ne.0)
        ind=ind+1
        if (ind.ne.inew(1)) then
          io=iold(ind)
          in=inew(i)
          iold(ind)=i
          inew(i)=ind
          iold(in)=io
          inew(io)=in
        endif
        i=next(1)
      enddo
      ind1=1
      ind2=ind
      ilasd=0

```

```

c
c         loop over levels
c

```

```

      do while (ind1.lt.ind2)

```

```

c         loop over nodes in level
c

```

```

      do in=ind1,ind2
        io=iold(in)
        ie=ieltn(io)
        is=isrc(io)
        if (is.eq.0) is=io

```



```

c      skipping active nodes.
c
      indsav = ind
      do while (ie.ne.0)
        k=(io-inode(1,ie))/(inode(2,ie)-inode(1,ie))
        jo=inode(2-k,ie)
        ie=ielte(k+1,ie)
        js=isrc(jo)
        if(js.ne.0)then
          jn=inew(jo)
          if (jn.gt.ind) then
            ind=ind+1
            indo=old(ind)
            old(ind)=jo
            inew(jo)=ind
            old(jn)=indo
            inew(indo)=jn
            isrc(jo) = is
          elseif (jn.gt.indsav) then
            isrc(jo) = is
          elseif (js.ne.is.and.jn.gt.in)then
            isrc(jo)=0
            next (ilasd)=jo
            ilasd=jo
          endif
          elseif (is.ne.jo.and.isrc(io).ne.0) then
            isrc(io) = 0
            is = io
            next (ilasd) = io
            ilasd = io
            ie = ieltn(io)
          endif
        enddo
      enddo
      ind1=ind2+1
      ind2=ind
    enddo
    next (ilasd)=0
c
c      downward search
c
      io=next(0)
      ilasf=0
c
c      loop over nodes as long as there are new ones to be screened
c
      do while (io.ne.0)
c
c        in=inew(io)
c        ie=ieltn(io)
c        isour=0
c
c        loop over nodes connected to node in by a line element
c
c        do while (ie.ne.0)
          k=(io-inode(1,ie))/(inode(2,ie)-inode(1,ie))
          jo=inode(2-k,ie)
          if (isrc(jo).ne.0)then
            jn=inew(jo)
            if(jn.le.in)then
c
c              flag node jn as active and mark to be
c              screened later in the downward search

```

```

        next(ilasd)=jo
        isrc(jo)=0
        ilasd=jo
        next(ilasd)=0
    else
c
c         flag node in to be marked as a source for
c         the next forward search
c
        isour=1
        endif
    endif
    le=leite(k+1,ie)
enddo
c
c     mark node as a source for next forward search
c
    if(isour.ne.0)then
        next(ilasf)=io
        ilasf=io
    endif
c
    io=next(io)
enddo
next(ilasf)=0
i=next(0)
enddo
c
c     take active nodes
c
nnodes=0
do io=1,nodeso
    in=new(io)
    if (isrc(io).eq.0) then
        nnodes=nnodes+1
        jo=old(nnodes)
        iold(in)=jo
        inew(jo)=in
        iold(nnodes)=io
        inew(io)=nnodes
    endif
enddo
do in=nnodes+1,nodeso
    io=iold(in)
    inew(io)=0
enddo
return
end

```

```

subroutine plot
common/forward/nnodes,neleme,maxd,newnel,nflow,ikeep,iplot
common/control/maele,maxnod
common/ieltn/ieltn(1)
common/iold/iold(1)
common/inew/inew(1)
common/ieite/ieite(2,1)
common/inode/inode(2,1)
common/ifr/ifr(2,1)
common/xyzphi/xyzphi(6,1)
character*11 file

```

c

```

10 write(file,10)nflow
   format('lines',12.2,'.dat')
   open(unit=7,file=file,status='unknown')
   do in=1,nnodes
     io=iold(in)
     ie=ieltn(io)
     do while (ie.ne.0)
       k=(io-inode(1,ie))/(inode(2,ie)-inode(1,ie))
       jo=inode(2-k,ie)
       jn=inew(jo)
       if (in.lt.jn) then
         x1=xyzphi(1,io)
         y1=xyzphi(2,io)
         z1=xyzphi(3,io)
         x2=xyzphi(1,jo)
         y2=xyzphi(2,jo)
         z2=xyzphi(3,jo)
         write (7,710) x1,y1,z1,x2,y2,z2,ifr(1,ie),ifr(2,ie)
         endif
         ie=ieite(k+1,ie)
       enddo
     enddo
   close (unit=7)
   return
710 format(6f10.4,2i5)
end

```

```

subroutine prout
common/ttitle/title
common/forward/nnodes, neleme, maxd, newnel, nflow, lkeep, lplot
common/control/maxele, maxnod
common/ieltn/ieltn(1)
common/iold/iold(1)
common/inew/inew(1)
common/ielts/ielts(2,1)
common/xyzphi/xyzphi(6,1)
common/ibs/ibs(2,1)
common/inode/inode(2,1)
common/aptdis/aptdis(2,1)
common/ifr/ifr(2,1)
common/ifrac/ifrac(1)
character*80 title(10)

```

c
c
c

```

compute bandwidth lbwth

```

```

lbwth=0
do in=1,nnodes
  io=iold(in)
  if (ibs(1,io).ne.1)then
    ie=ieltn(io)
    do while (ie.ne.0)
      k=(io-inode(1,ie))/(inode(2,ie)-inode(1,ie))
      jo=inode(2-k,ie)
      if (ibs(1,jo).ne.1)then
        jn=inew(jo)
        lbwth=max(jn-in,lbwth)
      endif
      ie=iwrite(k+1,ie)
    enddo
  endif
enddo
lbwth=lbwth+1

```

c

```

write (4,30) title,newnel,nnodes,maxd,lkeep,lplot,lbwth
do m=1,nnodes
  n=iold(m)
  if (ibs(2,n).eq.0) then
    write(4,40) m,ibs(1,n),ibs(2,n),ifrac(n),(xyzphi(1,n),1-1,3)
  else
    write(4,40) m,ibs(1,n),ibs(2,n),ifrac(n),(xyzphi(1,n),1-1,3+maxd)
  endif
enddo
m=0
do in=1,nnodes
  io=iold(in)
  ie=ieltn(io)
  do while (ie.ne.0)
    k=(io-inode(1,ie))/(inode(2,ie)-inode(1,ie))
    jn=inew(inode(2-k,ie))
    if (in.lt.jn) then
      m=m+1
      write (4,50) m,in,jn,aptdis(1,ie),aptdis(2,ie),
2         ifr(1,ie),ifr(2,ie)
    endif
    ie=ielts(k+1,ie)
  enddo
enddo
return

```

c

```

30 format(10(a/),
1  'nelements=',16,4x,'nnodes = ',16,4x,'nboundary-',16/

```

```
2  'trunc.code',15,5x,'plot code-',15,5x,'bandwidth-',15)
40 format(415,6f10.4)
50 format(315,5x,1p,2e10.4,0p,5x,215)
end
```

```

      subroutine rcntrl
c-----
c      imode= -2      Flow only (Steady-state, theta=1.0)
c      -1           (Transient)
c      0           Test data deck
c      1           Transport (Transient flow)
c      2           (Steady-state flow, theta=1.0)
c      ibmode= 0      Radial boundary (start from side 5 only)
c      1           Square boundary (start from side 2 only)
c      2           Square boundary (start from all four sides
c                      1, 2, 3, 4.)
c-----
      common /cntrl/  title2,imode,ibmode,mxstep,iuo,iui,
$                    nstep,time0,dtini,pr,r,tmax,theta,fsys,
$                    tole,dcint,dcon,dcoff,rhor,rmur,qavr,
$                    ssubs,dispc
      character       title2*80
c
      iui=3
      open (unit=iui,readonly,file='inter.inp',status='old')
      read(iui,501) title2
      read(iui,521) imode,ibmode,mxstep
c
      read(iui,511) time0,tmax,dtini
c
      read(iui,511) pr,r,theta,tle
c
      read(iui,511) dcint,dcon,dcoff
c
      read(iui,511) rhor,rmur,qavr
c
      read(iui,512) ssubs,dispc
c
      501 format (a80)
      511 format (3(10x,f10.0))
      512 format (2(10x,f10.0))
      521 format (3(10x,i5))
c
      return
      end

```

```

subroutine rdata
common/t title/t title
common/forward/nnodes, neleme, maxd, newnei, nflow, lkeep, lplot
common/control/maxele, maxnod
common/old/old(1)
common/new/new(1)
common/xyzphi/xyzphi(6,1)
common/lbs/lbs(2,1)
common/inode/inode(2,1)
common/aptdis/aptdis(2,1)
common/lfr/lfr(2,1)
common/lfrac/lfrac(1)
character*80 title(10)
character*11 file

```

```

c
c      read input data
c
      nflow=nflow+1
      write(file,10)nflow
10    format('renum',i2.2,'.dat')
c      open (unit=1,readonly,file=file,status='old',err=60)
      open (unit=1,file=file,status='old',err=60)
c
      read (1,70,end=10) title,neleme,nnodes,maxd,lkeep,lplot
      if (neleme .gt. maxele) then
        write(6,*) maxele
        call werror('neleme .gt. maxele')
      endif
      if (nnodes .gt. maxnod) then
        write(6,*) maxnod
        call werror('nnodes .gt. maxnod')
      endif
      if (neleme .lt. 1) then
        write(6,*) neleme
      elseif (nnodes.gt.0) then
c
        read(1,80) (lbs(1,m),lbs(2,m),lfrac(m),
2          (xyzphi(1,m),i=1,6),m=1,nnodes)
        read(1,90) (inode(1,n),inode(2,n),aptdis(1,n),aptdis(2,n),
1          lfr(1,n),lfr(2,n),n=1,neleme)
        endif
        return
60    stop
c
70    format(10(a/),3(10x,i6,5x)/2(10x,15,5x))
80    format(5x,3i5,6f10.0)
90    format(5x,2i5,5x,2e10.4,5x,2i5)
      end

```

```

subroutine rnpn
c
common /ibs/ ibs(2,1)
logical*1 fx4
c
inquire(file='npn.inp',exist=fx4)
if(.not.fx4) return
open(unit=12,file='npn.inp',status='old')
1 read(12,700,end=99) listp
if(ibs(1,listp).eq.0) then
    ibs(1,listp)--99
else
    ibs(1,listp)--abs(ibs(1,listp))
endif
700 format(15)
go to 1
99 close(unit=12)
return
end

```



```

subroutine triout
common/title/title
common/forward/nnodes,neleme,maxd,newnei,nflow,ikeep,iplot
common/control/maxele,maxnod
common/ieltn/ieltn(1)
common/iod/iod(1)
common/inew/inew(1)
common/ielte/ielte(2,1)
common/xyzphi/xyzphi(6,1)
common/lbs/lbs(2,1)
common/inode/inode(2,1)
common/aptdis/aptdis(2,1)
common/lfr/lfr(2,1)
common/lfrac/lfrac(1)
common /cgeo/   dc(1),ss(1),w(1)
common /ctrl/   title2,inode,lbmode,mxstep,luo,lui,
$               nstep,time0,dtini,prf,tmax,theta,fsys,
$               tole,dcint,dcon,dcoff,rhor,rmur,gravr,
$               ssubs,dispc
character*80 title(10),title2

```

```

c
cube = 1./3.
open(unit=8,status='unknown',file='ctrl.inp',
$   carriagecontrol='list')
open(unit=9,status='unknown',file='node.inp',
$   carriagecontrol='list')
open(unit=10,status='unknown',file='elmt.inp',
$   carriagecontrol='list')
write(8,620) title(1),title(2),title2
write(8,621) inode,lbmode,mxstep
write(8,631) time0,tmax,dtini
write(8,641) prf,theta,tle
write(8,651) dcint,dcon,dcoff
write(8,661) rhor,rmur,gravr
write(9,670) nnodes,title(1)

c
c compute bandwidth lbwth
c
lbwth=0
do in=1,nnodes
  io=iod(in)
  if (lbs(1,io).ne.1)then
    ie=ieltn(io)
    do while (ie.ne.0)
      k=(io-inode(1,ie))/(inode(2,ie)-inode(1,ie))
      jo=inode(2-k,ie)
      if (lbs(1,jo).ne.1)then
        jn=inew(jo)
        lbwth=max(jn-in,lbwth)
      endif
      ie=ielte(k+1,ie)
    enddo
  endif
enddo
lbwth=lbwth+1
print *, 'bandwidth =',lbwth

c
lbc = 0
do m=1,nnodes
  n = iold(m)
  if (xyzphi(3,n).eq.0.and.xyzphi(4,n).eq.0.) then
    write(9,674) m,lbs(2,n),lbs(1,n),lbc,(xyzphi(j,n),j=1,2)
  elseif (xyzphi(4,n).eq.0.) then
    write(9,673) m,lbs(2,n),lbs(1,n),lbc,(xyzphi(j,n),j=1,3)
  endif
enddo

```

```

    else
      write(9,671) m, lbs(2,n), lbs(1,n), lbc, (xyzphi(j,n), j=1,4)
    endif
  end do
  write(10,680) newnel, title(1)
  m=0
  do in=1,nnodes
    io=ioid(in)
    ie=ieitn(io)
    do while (ie.ne.0)
      k=(io-lnode(1,ie))/(lnode(2,ie)-lnode(1,ie))
      jn=lnew(lnode(2-k,ie))
      if (in.lt.jn) then
        m=m+1
      c! currently, the cubic law is assumed
        w(ie)=(aptdis(1,ie)**12.*rmur/rhor/gravr)**cube
      c! cu. vently, Ss and D are constant value
        ss(ie)=ssubs
        dc(ie)=dispc
        if (dispc.eq.0.) then
          write(10,682) m, in, jn, aptdis(1,ie), w(ie),
          $          aptdis(2,ie), ss(ie)
        else
          write(10,681) m, in, jn, aptdis(1,ie), w(ie),
          $          aptdis(2,ie), ss(ie), dc(ie)
        endif
      endif
      ie=leite(k+1,ie)
    enddo
  enddo
c
  close(unit=7)
  close(unit=8)
  close(unit=9)
  return
c
620 format(a)
621 format('lnode      ',15,'lnode      ',15,'mstep      ',15)
631 format('time0      ',1p10.4,'tmax      ',1p10.4,
$        'dtini      ',1p10.4)
641 format('prr      ',1p10.4,'theta      ',1p10.4,
$        'tote      ',1p10.4)
651 format('dcint      ',1p10.4,'dcon      ',1p10.4,
$        'dcoff      ',1p10.4)
661 format('rhor      ',1p10.4,'rmur      ',1p10.4,
$        'gravr      ',1p10.4)
670 format('an      ',15,1x,a)
671 format(15,13,2i2,5f12.4)
673 format(15,13,2i2,3f12.4)
674 format(15,13,2i2,2f12.4)
680 format('ne      ',15,1x,a)
681 format(3i5,1p,5e12.4)
682 format(3i5,1p,4e12.4)
c
end

```

```

      subroutine weerror(line)
c
c
      character*(*) line
c
      open (unit=11,file='sys$error',access='append',status='unknown')
      write (11,100) line
      close (unit=11)
      stop
100 format(' error ',a)
      end

```

Appendix E

LINEL - Program Organization and Arrays

Table E-1. LINEL - Description of Program Variables

Variable	Description	How Value is Assigned
ibwth	Bandwidth of the linear system	Read by RHEAD
idate	Current date, character*9	Call to the date subroutine
igrad	Type of gradient to be used for study region permeability calculations = 0 use both types of gradients = 1 use global gradient only = 2 use local gradient only	Read by RSTUD
impfix	Flag for the presence of imposed flux nodes	Set in RMESH
ipmt	Code for printing output = 0 print normal output in LINEL.OUT = 1 print normal output plus heads at disc intersections in LINEL.OUT (3D only) = 2 print detailed output in LINELALL.OUT	Read LINEL.INP
istud	Number of study regions	Read by RSTUD
i8open	Flag which says if LINELALL.OUT is open or not	Subroutine RDATA
jdate	Character variable containing the date at which the line network was generated	Read from LINEL.INP
jobnam	Character variable identifying the program that generated the line network	Read from LINEL.INP
maxd	Number of different boundary conditions to be solved for the same network	Read by RHEAD
maxele	Maximum number of elements used to check the size of the problem	Set in main
maxnod	Maximum number of nodes used to check the size of the problem	Set in main

mxelem	Maximum number of elements used to dimension arrays	Parameter statement in main
mxnode	Maximum number of nodes used to dimension arrays	Parameter statement in main
neleme	Number of elements in the network	Read by RHEAD
nfrac	Number of fractures in the flow region	Read by RHEAD
ngrad	Index for the do loops for global and local gradients	Used in KSTUD and SFLUX
nnodes	Number of nodes in the network	Read by RHEAD
noddim	Dummy parameter in subroutines (noddim = maxnod)	
nside	Number of sides, depending on the type of problem (2-dimensional rectangle or circle, or 3-dimensional)	Set in RHEAD
rotan rotan2	Rotation angles of the flow region (rotan2 = 0 for 2-D cases) For circular flow region, rotan and rotan2 are the coordinates of the center of the hole	Read by RHEAD
spgr	Specific gravity in the unit system chosen by the user	Read by RHEAD
visc	Dynamic viscosity of water in the unit system chosen by the user	Read by RHEAD
xgene,ygene,zgene	Size of the generation region (zmesh = 0 for 2-D cases) for circular generation regions, y-gene = 0, and the radius is read in xgene	Read by RDATA
xmesh,ymesh,zmesh	Size of the flow region (z mesh = 0 for 2-D cases) for circular flow regions, xmesh = radius of flow region, ymesh = radius of hole	Read from RDATA

Table E-2. LINEL - Description of Program Arrays

Array	Description	How Value is Assigned
a(ndima)	Matrix representing the linear system, stored in banded form. a(i) = an element of the matrix Note that a is passed as a two-dimensional routine to subroutines CPHI and SYMSOL	Built by CPHI
b(mxnode,3)	Right-hand side vector of the linear system, contains the solution after SYMSOL. i = 1 nnodes; irot = 1, maxd b(i,irot) = value of the right-hand side for line i of the linear system under boundary conditions number irot (irot = 1 or 2 for two-dimensional cases)	Build by CPHI, modified by SYMSOL
bvalu(i),i=1,6	Values of the boundary head or flux on side i of the flow region (bvalu(5) = bvalu(6) = 0 for 2-D cases)	Read by RDATA
coord(3,mxnode)	node coordinates; n=1 nnodes coord(1,n) = x coordinate of node n coord(2,n) = y coordinate of node n coord(3,n) = z coordinate of node n (0 if 2-D)	Read by RDATA
dist(mxelem)	Element length; ie=1 neleme dist(ie) = length of element number ie	Read by RDATA
flx(mxelem)	Flux solution vector; ie = 1, neleme flx(ie) = flux in element ie	Computed by CUNK
ibcode(6)	The boundary code for side i of the flow region (ibcode(5) = ibcode(6) = 0 for 2-D cases) = -1 - constant flux = 0 - internal node = 1 - constant head = 2 - constant linearly distributed head	Read by RDATA
ib(mxnode)	Node type; n=1, nnodes ib(n) = 0 internal node = 1 imposed head boundary node = -1 imposed flux boundary node	Read by RDATA

inode(2,mxelem)	Element-node reference array; ie = 1,neleme inode(1,ie) = number of the node making up the first endpoint of element number ie inode(2,ie) = number of the node making up the second endpoint of element number ie	Read by RDATA
phi(mxnode,3)	Imposed head or flux if relevant, and then head solution vector; n = 1, nnodes irot = 1,maxd phi(n,irot) = 0 if node n is internal phi(n,irot) = value of imposed head or flux phi(n,irot) = head computed or imposed at node n	Read by RDATA Assigned by CUNK
side(mxnode)	Boundary side number, n=1, nnodes side(n) = side on which boundary node is lying = 0 if internal node	Read by RDATA
sgrad(2)	Character array which stores "g" (standing for global gradient) and "l" (standing for local gradient)	
title(2)	The title of the problem to be solved. Character*80. Read from LINEL.INP.	Read by RDATA
trans(mxelem)	Element transmissivities array; ie = 1, neleme trans(ie) = transmissivity of element number ie	Read by RDATA
xstud(20)	Size of the study regions in the x direction	Read by main
ystud(20)	Size of the study regions in the y direction	Read by main

Table E-3. LINEL - Subroutine Outline

LINEL	RSTUD	
	RHEAD	WERROR
	RMESH	
	WERROR	
	SORTIS	
	CPHI	WERROR
	SYMSOL	
	CUNK	
	PINFO	
	SFLUX	
	STUDY	

Table E-4. LINEL - Description of Subroutines

Subroutine	Description
CPHI	<p>Fills the matrix and the right-hand side vector b.</p> <p>given: flx(nnodes), ib(nnodes), inode(2,neleme), nnodes, neleme, idima, ibwth, maxd, phi(nnodes,maxd), dist(neleme), coord(3,nnodes), trans(nnodes)</p> <p>returns: a(ndima), b(nnodes,maxd)</p>
CUNK	<p>Transfers the solution to the head vector phi, and computes the flux through each element.</p> <p>given: nnodes, neleme, b(nnodes,maxd), inode(2,neleme), ib(nnodes), trans(neleme), dist(neleme)</p> <p>returns: phi(nnodes,maxd), flx(neleme)</p>
PINFO	<p>Prints the headers for the output files and the "dump" file if specified</p> <p>given: nside, iray, idate, jobnam, date, title, nfrac, ibwth, xgene, ygene, zgene, xmesh, ymesh, zmesh, rotan, rotan2, neleme, nnodes, ibcode(6), bvalu(6) bound(6,3), iskip8, spgr, visc, inode(2,neleme), trans(neleme), dist(neleme), phi(nnodes,maxd),</p> <p>writes: part or all of the above, depending on iskip8</p>
RHEAD	<p>Reads the header of the input file LINEL.INP (group 2, Table 4-1)</p> <p>reads: all input from group 2 (file LINEL.INP) as specified in Table 4-1</p> <p>returns: everything it reads plus nside</p>
RMESH	<p>Reads the nodes and elements making up the network (group 3 in Table 4-1)</p> <p>given: all the parameters read by RHEAD (group 2 in Table 4-1)</p> <p>reads: nodes and elements definition (group 3 in Table 4-1)</p> <p>returns: everything it reads, plus flx (nnodes), maxd, impflx</p> <p>writes: everything read by RHEAD or itself on a dump file named LINEL.ALL.OUT, depending on parameter iskip8</p>

RSTUD	Reads study regions data (group 1 in Table 4-1) reads: input group 1 returns: what it reads, except ngrad which is modified
SFLUX	Computes and prints out the flux through each side. given: istud, igrad, iray, idate, jobnam, date, title(2) xstud(istud), ystud(istud), flx(neleme), iskip8, rotan, irot, nside, xmesh, ymesh, zmesh writes: total fluxes through the two, four or six boundaries, depending on nside; number of fractures per unit area or per unit length on each boundary
SORTU	Sorts elements, and nodes within elements, to insure that the two nodes in an element are in increasing order, and that the elements are in non-decreasing order with regard to their first node. given: neleme, inode(2,neleme), trans(neleme), dist(neleme) returns: modified inode, trans, dist
STUDY	Looks for intersections between fractures and the study region boundaries. Then computes the average head at these boundaries, and the flux through these bound- aries. given: rotan, istud, neleme, nnodes, inode(2,neleme), coord(3,nnodes), xstud(istud), ystud(istud), trans(neleme), phi(nnodes) writes: equivalent permeability for each direction of each study region
SYMSOL	Linear equation solver given: ibwth, a(ndima), nnodes, b(nnodes, maxd) returns: modified b containing the solution of the linear system
WERROR	Prints out error messages on file LINEL.ERR given: line prints it.

Appendix F

LINEL - Program Listing

```

program line1
implicit real*8 (a-h,o-z)

```

```

      version 3.0 (Jan 1986)
      this program is used to calculate the flux
      through line elements

```

```

      a = the coeff matrix

```

```

      title = the title of the problem to be solved

```

```

      iray = array to store the jobcard information

```

```

      ibcode = the boundary code

```

```

      -1 = flux specified

```

```

      0 = internal node

```

```

      1 = phi specified

```

```

      bvali = code used to determine the value of phi
              on the boundary

```

```

      idate = variable to store the date of program run

```

```

      jobnam = the name of the job

```

```

      jdate = the date of creation of input to line1

```

```

      inode = the node numbers for the element

```

```

      trans = the transmissivity of the element

```

```

      coord = coordinates of the node

```

```

      side = the side on which the node is located

```

```

      phi = the potential of the node

```

```

      flx = net flux into or out of node n

```

```

      0 = an internal node

```

```

      b = the solution array

```

```

parameter (mxnode=75000,mxelem=85000,ndima=6000000)

```

```

common/trans/ trans(mxelem)

```

```

common/bcode/ ibcode(6),bvali(6)

```

```

common/coord/ coord(3,mxnode)

```

```

common/dist/ dist(mxelem)

```

```

common/flux/ flx(mxnode)

```

```

common/lb/ lb(mxnode)

```

```

byte lb

```

```

common/inode/ inode(2,mxelem)

```

```

common/lfrac/ lfrac(2,mxnode)

```

```

common/mesh/ xmesh,ymesh,zmesh,xgene,ygene,zgene,

```

```

1      rotan,rotan2,nfrac

```

```

common/param/ maxele,maxnod,nelame,nnodes,iprint,maxd,igrad,

```

```

1      nside,impflx

```

```

common/pi/ pi180

```

```

common/side/ side(mxnode)

```

```

byte side

```

```

common/stud/ istud,xstud(20),ystud(20)

```

```

c
c      array a is a two-dimensional array of columns by rows,
c      however the number of columns is unknown until the bandwidth
c      has been calculated

```

```

      dimension a(ndima),b(mxnode,2),phi(mxnode,2),lnew(mxnode)

```

```

      common/title/ title(2)

```

```

      character*80 title

```

```

      common/titles/iray,irate,jobnam,jdate

```

```

      character*7 iray,jobnam

```

```

      character*9 idate,jdate

```

```

      common/vispg/ visc,spqr

```

```

c
c      calculate pi/180 once

```

```

      pi180=atan(1.d0)/45.d0

```

```

      maxele=mxelem

```

```

      maxnod=maxnode
c
c      iray = 'line1'
      call date(idate)
c
c
c if maxd is 2 line1 will attempt to find a second column in
c the b vector and calculate fluxes for rotan=90.
c if lb(iside) is .ie. 0, maxd is set to 1 in rdata
c
      maxd=2
c
c read data for study region:
c
      call rstud
c
c      open(unit=1,file='line1.inp',readonly,status='old')
      open(unit=6,file='line1.out',status='unknown')
      open(unit=11,file='ellipse.inp',status='unknown',
1         carriagecontrol='list')
10 call rhead(lbuth)
   if (nnodes.lt.0) stop
   call rmesh(phi,maxnod,lbuth,lnew,na)
   if (na*lbuth.gt. ndima) call werror('array a is too small-lin')
c
c   if (nnodes.ne.0) then
c     call sort1)
c
c     start=cputime(0.0)
c
c     call cphi(lbuth,na,a,maxnod,b,phi,lnew)
c     call symsol(lbuth,na,a,maxnod,b)
c     endif
c
c     aver = 0.d0
c$dir scalar
c     do irot=1,maxd
c       if(na.ne.0)then
c
c         call cuak(phi(1,irot),b(1,irot),lnew)
c         endif
c
c         call pinfo(phi(1,irot),b(1,irot),lbuth,irot,lnew)
c         call s`ux(phi(1,irot),b(1,irot),irot,lnew)
c
c         if (lstud.ne.0) call study (phi(1,irot))
c         rotan = rotan + 90.d0
c       enddo
c       time=cputime(0.0)-start
c       write(6,20)time
20 format('/// Time spent filling the matrix and solving the/'
1      ' linear system (in cpu seconds):',f15.2///)
c       go to 10
c
c     end

```

```

subroutine cphi(ibwth,na,a,noddim,b,phi,inew)
implicit real*8 (a-h,o-z)

```

```

c
c      version 4.0 ( Dec 1987 )
c

```

```

c *** method of solution to [a] x = [b]
c

```

```

c *** if node(i) is prescribed, the inew(i)-th row and
c *** column of [a] are set to zero and, hence, contributes
c *** nothing to the solution, placing these nodes last (using
c *** inew) allows us to omit them from the matrix solver and
c *** still leave them in b. b(inew(i)) is replaced by
c *** the known value of x(i). each of the affected
c *** terms of b is modified by subtracting from it the value
c *** of the prescribed nodal variable multiplied by the
c *** appropriate column term from the original [a] matrix.
c

```

```

c
common/trans/ trans(1)
common/coord/ coord(3,1)
common/dist/ dist(1)
common/flux/ flx(1)
common/ib/ ib(1)
byte ih
common/inode/ inode(7,1)
common/param/ maxele,maxnod,nelems,nnodes,iprint,maxd,igrad,
1 nside,ispflx
common/side/ side(1)
byte side
common/vispg/ visc,spgr
dimension phi (noddim,2),b(noddim,2),inew(*)
dimension a(ibwth,na)

```

```

c
c *** initialize a & b
c

```

```

c the following loop is to initialize the entire a array
c it is done in this manner to prevent the compiler from
c vectorizing just the column stores (ibwth direction)
c

```

```

do i = 1, ibwth*na
  a(i,1) = 0.d0
enddo
do j=1,nnodes
  b(j,1)=0.0d0
  b(j,2)=0.0d0
enddo
do j=1,nnodes
  if (ib(j).lt.0) b(inew(j),1)=flx(j)
enddo

```

```

c
c calculate banded coefficient matrix and modify [b]
c fill banded matrix [a] with lower triangle values where:
c number of rows = number of non-boundary nnodes
c number of columns = bandwidth
c and the first column is the diagonal
c and the other columns are the non-zeros off-diagonals
c of the coefficient matrix
c

```

```

c
izero = 0
do n=1,neleme
  if (dist(n).eq.0.d0) then
    write(6,70) n
    izero = 1

```

```

      if (lzero.ne.0) then
        call werror('there is an element with zero length')
        return
      endif
      do n=1,nelems
        n1 = inode(1,n)
        n2 = inode(2,n)

c
c *** calculate element term (e) of coefficient matrix
c
        e=trans(n)/dist(n)

c
c
        if (ib(n1).ne.1) then
          a(1,inew(n1)) = a(1,inew(n1)) + e
          if (ib(n2).ne.1) then
            a(1,inew(n2)) = a(1,inew(n2)) + e
c
c *** head not prescribed at either node of element so put
c *** e into its off diagonal position in band storage
c      save storage in [z] for all diagonal elements
c
            a(inew(n2)-inew(n1)+1,inew(n1)) = -e
            else
c
c *** else if node n2 prescribed head modify [b] only
c
c$dir scalar
            do irot=1,maxd
              b(inew(n1),irot)=b(inew(n1),irot)+e*phi(n2,irot)
            enddo
            endif
            elseif (ib(n2).ne.1) then
              a(1,inew(n2)) = a(1,inew(n2)) + e
c
c *** else if node n1 prescribed head modify [b] only
c
c$dir scalar
            do irot=1,maxd
              b(inew(n2),irot)=b(inew(n2),irot)+e*phi(n1,irot)
            enddo
            endif
            enddo
c
c      put the prescribed-head into [b].
c      (this is the one place where iold would have been useful)
c
c$dir scalar
      do irot = 1,maxd
        do ir=1,nnodes
          if (ib(ir).eq.1) b(inew(ir),irot) = phi(ir,irot)
        enddo
      enddo
      return
70 format('element ',15,' has zero length, try cphi2')
end

```



```

      subroutine cunk(phi,b,inew)
      implicit real*8 (a-h,o-z)

c
c      version 3.0 (jan 1986)
c      this subroutine calculates values of phi
c
      common/trans/ trans(1)
      common/coord/ coord(3,1)
      common/dist/  dist(1)
      common/flux/  flux(1)
      common/lb/    lb(1)
      byte lb
      common/inode/ inode(2,1)
      common/side/  side(1)
      byte side
      common/param/ maxele,maxnod,nelene,nnodes,iprint,maxd,igrad,
1      nside,impflx
      common/vispg/ visc,spgr
c      only the column of phi and b for this rotation are passed
      dimension phi(1),b(1),inew(1)
c
      do i = 1, nnodes
         phi(i) = b(inew(i))
      enddo
c
c      calculate flux of constant head boundary nodes
c
      do i=1,nnodes
         if (lb(i) .eq. 1) flux(i) = 0.d0
      enddo
      do i = 1, nelene
         n1 = inode(1,i)
         n2 = inode(2,i)
         if (lb(n1) .eq. 1)
1      flux(n1) = flux(n1) + trans(i)*(phi(n1)-phi(n2))/dist(i)
         if (lb(n2) .eq. 1)
1      flux(n2) = flux(n2) + trans(i)*(phi(n2)-phi(n1))/dist(i)
      enddo
      return
      end

```

```

subroutine plinfo(phi,b,ibwth,irot,inew)
implicit real*8 (a-h,o-z)

```

```

version 3.0 (jan 1986)

```

```

common/trans/ trans(1)
common/bcode/ ibcode(6),bvalu(6)
common/coord/ coord(3,1)
common/dist/ dist(1)
common/flux/ flx(1)
common/ib/ ib(1)
byte ib
common/inode/ inode(2,1)
common/ifrac/ ifrac(2,1)
common/mesh/ xmesh,ymesh,zmesh,xgene,ygene,zgene,
1 rotan,rotan2,nfrac
common/param/ maxele,maxnod,neleme,nnodes,iprint,maxd,igrad,
1 nside,impflx
common/side/ side(1)
byte side
common/title/title(2)
character*80 title
common/titles/iray,ldate,jobnam,date
character*7 iray,jobnam
character*9 ldate,date
common/vispg/ visc,spgr
dimension phi(1),b(1),inew(1)
dimension boun(6,3)
data boun/ 1.d0, -1.d0, 0.d0, -1.d0, -1.d0, -1.d0,
1 -1.d0, -1.d0, -1.d0, -1.d0, 1.d0, 0.d0,
2 -1.d0, 1.d0, -1.d0, 0.d0, -1.d0, -1.d0/

```

```

if (nside.eq.2) then
write(6,80) iray,ldate,jobnam,date,title,nfrac,ibwth,
1 xgene,xmesh,ymesh,rotan,rotan2,neleme,nnodes
ibc=ibcode(1)
if (ibc.eq.-1) then
write(6,210)bvalu(1)
elseif (ibc.eq.1) then
write(6,220)bvalu(1)
endif
ibc=ibcode(3)
if (ibc.eq.-1) then
write(6,230)bvalu(2)
elseif (ibc.eq.1) then
write(6,240)bvalu(2)
endif

```

```

if (iprint.lt.2) then
if (impflx.ne.0) goto 1000
return
endif
elseif (nside.eq.4) then
write(6,90) iray,ldate,jobnam,date,title,nfrac,ibwth,
1 xgene,ygene,xmesh,ymesh,rotan,neleme,nnodes

```

```

c$dir scalar

```

```

do i=1,nside
ibc = ibcode(i)
if (ibc .eq. -1) then
write(6,110) i,bvalu(i)
elseif (ibc .eq. 0) then
write(6,120) i

```

```

        write(6,130) i,bvalu(i)
        elseif (ibc .eq. 7) then
            write(6,140) i
        endif
    enddo
c
cccc write on tape 8 if iprint is 2
c
    if (iprint.lt.2)then
        if (impfix.ne.0)goto 1000
        return
    endif
c
    write(8,90) iray,ldate,jobnam,date,title,nfrac,lbwth,
1      xgene,ygene,xmesh,ymesh,rotan,neleme,nnodes
c
c$dir scalar
    do i=1,nside
        ibc = ibcode(i)
        if (ibc .eq. -1) then
            write(8,110) i,bvalu(i)
        elseif (ibc .eq. 0) then
            write(8,120) i
        elseif (ibc .eq. 1) then
            write(8,130) i,bvalu(i)
        elseif (ibc .eq. 2) then
            write(8,140) i
        endif
    enddo
c
    else
        write(6,100) iray,ldate,jobnam,date,title,nfrac,lbwth,
1      xgene,ygene,zgene,xmesh,ymesh,zmesh,
2      rotan,rotan2,neleme,nnodes
c
        ibc = ibcode(irot)
c$dir scalar
        do i=1,nside
            if(boun(i,ibc).ge.0.)then
                write(6,130) i,bvalu(irot)*boun(i,ibc)
            else
                write(6,140) i
            endif
        enddo
c
cccc write on tape 8 if iprat is 2.
c
    if (iprint.lt.2)then
        if (impfix.ne.0)goto 1000
        return
    endif
c
    write(8,100) iray,ldate,jobnam,date,title,nfrac,lbwth,
1      xgene,ygene,zgene,xmesh,ymesh,zmesh,
2      rotan,rotan2,neleme,nnodes
c
c$dir scalar
    do i=1,nside
        if(boun(i,ibc).ge.0.)then
            write(8,130) i,bvalu(irot)*boun(i,ibc)
        else
            write(8,140) i
        endif
    enddo

```

```

endif
c
c calculate flux and velocity in each fracture
c
write(8,150)
c
qc=spqr/(12.d0*visc)
do i = 1, neleme
  n1 = inode(1,i)
  n2 = inode(2,i)
  fr = trans(i)*(phi(n1) - phi(n2))/dist(i)
  re = fr/visc
  aper = (trans(i)/qc)**(1.d0/3.d0)
  vel= fr/aper
  write(8,160) i,aper,vel,fr,re
enddo
c
write(8,170)
do i = 1, nnodes
  write(8,181) i,phi(i),flx(i),ifrac(1,i),ifrac(2,i), ! jep 1jul88
1  (coord(j,i),j=1,3),side(i) ! add coord
enddo
c
if(impflx.eq.0)return
c
1000 write(6,185)
do i = 1, nnodes
  if(ib(i).lt.0)then
    write(6,181)i,phi(i),flx(i),ifrac(1,i),ifrac(2,i), !jep 1jul88
1  (coord(j,i),j=1,3),side(i) !add coord
    endif
  enddo
  write(6,190)
c
return
c
80 format(/// --- ',a7,' --- ',a9//
1 ' Jobname and date of finite element mesh creation --- ',a7,
2 ', ',a9//1x,a80/1x,a80//
3 ' the number of fractures in the flow region is',i8/
4 ' the bandwidth for the matrix is',17x,i5/
5 ' the radius of the generation region is',8x,f7.2/
6 ' the radius of the flow region is ',8x,f7.2/
7 ' the radius of the inner hole is ',8x,f7.2/
8 ' the coordinates of the hole are x=',8x,f10.4,' and y=',f10.4//
8 ' neleme = ',15,10x,'nnodes = ',15)
90 format(/// --- ',a7,' --- ',a9//
1 ' Jobname and date of finite element mesh creation --- ',a7,
2 ', ',a9//1x,a80/1x,a80//
3 ' the number of fractures in the flow region is',i8/
4 ' the bandwidth for the matrix is',17x,i5/
5 ' the size of the generation region is',f7.2,' by',f7.2/
6 ' the size of the flow region is ',f7.2,' by',f7.2/
7 ' the angle of rotation is',19x,f10.2//
8 ' neleme = ',15,10x,'nnodes = ',15)
100 format('1 --- ',a7,' --- ',a9//
1 ' Jobname and date of finite element mesh creation --- ',a7,
2 ', ',a9//1x,a80/1x,a80//
3 ' the number of fractures in the flow region is',i8/
4 ' the bandwidth for the matrix is',17x,i5/
5 ' the size of the generation region is',f7.2,2(' by',f7.2)/
6 ' the size of the flow region is ',f7.2,2(' by',f7.2)/
7 ' the rotation angles phi and theta are',f4.0,' and ',f5.0//
8 ' neleme = ',15,10x,'nnodes = ',15)

```

```

110 format(' side ',i1,' is a constant flowrate boundary with value',
1      ' - ',f10.4)
120 format(' side ',i1,' is a no flow boundary')
130 format(' side ',i1,' is a constant pressure boundary',
1      ' with value = ',f10.4)
140 format(' side ',i1,' is a constant pressure boundary' ,
1      ' - the pressure varies linearly')
150 format('Element number fracture width(cm) velocity(cm/sec)'
1      ' flow rate(cc/sec) reynolds number')
160 format(i1,i10,i1e24.5,i1p3e20.5)
170 format('Node number head(cm) flow rate(cc/sec) side')
180 format(i1,i17,i1p2e17.5,6x,i1)
181 format(i1,i16,2e13.5,2i6,3f10.4,i3)
185 format('Imposed flux node(s)')
1      ' node number head flow rate side')
190 format(')
210 format('The center hole has an imposed flux of ',f10.4)
220 format('The center hole has an imposed head of ',f10.4)
230 format('The outer circle has an imposed flux of ',f10.4)
240 format('The outer circle has an imposed head of ',f10.4)
end

```

```

subroutine rhead(ibuth)
implicit real*8 (a-h,o-z)

```

```

version 3.0 (jan 1986)

```

```

this subroutine reads in the header of the data file

```

```

common/trans/ trans(1)
common/bcode/  ibcode(6),bvalu(6)
common/coord/  coord(3,1)
common/dist/   dist(1)
common/flux/   flx(1)
common/ib/     ib(1)
byte ib
common/inode/  inode(2,1)
common/mesh/   xmesh,ymesh,zmesh,xgene,ygene,zgene,
1             rotan,rotan2,nfrac
common/param/  maxele,maxnod,nlele,nnodes,iprnt,maxd,igrad,
1             nside,impfix
common/side/   side(1)
byte side
common/title/  title(2)
character*80 title
common/titles/ iray, idate, jobnam, date
character*7 iray, jobnam
character*9 idate, date
common/vispg/  visc, spgr
data i@open/0/

```

```

read input data

```

```

read(1,200,end=190) jobnam,date,iprnt,title,nfrac,
1             xgene,ygene,zgene,xmesh,ymesh,zmesh,rotan,
2             rotan2,ibcode,bvalu,visc,spgr,nlele,nnodes,
3             maxd,ibuth
if (nlele .gt. maxele) then
write(6,*) maxele
call werror('nlele .gt. maxele')
endif
if (nnodes .gt. maxnod) then
write(6,*) maxnod
call werror('nnodes .gt. maxnode')
endif
if (ygene.eq.0)then
nside=2
elseif(zgene.eq.0.)then
nside=4
else
nside=6
endif

```

```

return

```

```

190 stop

```

```

200 format(a19,3x,a9,11/2(a/)10x,15/2(3(10x,f10.4)/),2(10x,f10.4)/
1 10x,6110/10x,6110.0/2(10x,f10.0)/3(10x,16,4x)/50x,15)
end

```

```

subroutine rmesh(phi,noddim,ibwth,inew,na)
implicit real*8 (a-h,o-z)

c
c      version 3.0 (jan 1986)
c
c      this subroutine reads in the input data
c
common/trans/ trans(1)
common/bcode/  bcode(6),bvalu(6)
common/coord/  coord(3,1)
common/dist/   dist(1)
common/flux/   flx(1)
common/ib/     ib(1)
byte ib
common/inode/  inode(2,1)
common/ifrac/  ifrac(2,1)
common/mesh/   xmesh,ymesh,zmesh,xgene,ygene,zgene,
1  rotan,rotan2,nfrac
common/param/  maxele,maxnod,nelene,nnodes,iprint,maxd,igrad,
1  nside,impflx
common/side/   side(1)
byte side
common/title/ title(2)
character*80 title
common/titles/iray,ideate,jobnam,date
character*7 iray,jobnam
character*9 ideate,date
common/vispg/  visc,spgr
dimension phi (noddim,2),inew(noddim)
data isopen/0/

c
c  set up indices for in & out of matrix
  lim = 0
  lom = nnodes
  if (nnodes.ge.1) then
c
    do i = 1, nnodes
      read(1,210) ib(1),side(1),(coord(j,1),j=1,3),
1      (phi(1,1rot),1rot=1,maxd)
    enddo
c
c  check to see that a 90 deg rotation is appropriate
c  if all boundaries are not constant head, the a matrix will
c  be different, also set inew to record whether or not this
c  node will be recorded in the matrix.
c
    do i = 1, nnodes
      if (ib(1).le.0) then
        lim = lim + 1
        inew(i) = lim
        flx(1) = phi(1,1)
        if (ib(i).lt.0) impflx=1
        if (side(i).ne.0) maxd = 1
      else
        inew(i) = lom
        lom = lom - 1
      endif
    enddo
    endif
    na=lim

c
    if (nelene .lt. 1) then
      write(6,*) nelene
    else

```

```

c *****
c jep 1jul88 - change boundary cond. to print out fracture intersection heads
c
      do n = 1,neleme
        read(1,220) inode(1,n),inode(2,n),trans(n),dist(n),if1,if2
        if((if1.ne.if2.and.iprnt.eq.1) then
          do i1 = 1,2
            in0 = inode(i1,n)
            if((ib(in0).le.0) then
              ib(in0) = -1
              if(i1.eq.1) ifrac(i1,in0) = if1
              if(i1.eq.2) ifrac(i1,in0) = if2
              impflx = 1
            end if
            in00 = in0
          end do
          write(30,*) ifrac(1,in00),ifrac(2,in00)
        end if
      end do
    endif

c
c      write output information on tape8 if iprnt=2
c
      if (iprnt.lt.2) return
      if (i8open.eq.0) open (unit=8,file='line1all.out',
1 status='unknown',carriagecontrol='list')
      i8open=1
      write(8,230) iray,ldate,jobnam,date,title,nfrac,
1      xgene,ygene,zgene,xmesh,ymesh,zmesh,rotan,
2      rotan2,ibcode,bvalu,visc,spqr,nnodes

c
      write(8,240) (m,ib(m),side(m),(coord(1,m),i=1,3),
1      phi(m,1),flx(m),m=1,nnodes)
      write(8,250) neleme

c
      write(8,260) (n,inode(1,n),inode(2,n),trans(n),dist(n),n=1,neleme)

c
      return
c
210 format(5x,2i5,5x,6f10.4)
220 format(5x,2i5,5x,2e10.4,5x,2i5)
230 format('1 --- ',a7,' --- ',a9//' input data'//
1 ' jobname and date of finite element mesh creation ---',a7,
2 ' ',a9/lx,a80/lx,a80//
3 ' nfrac =',i5/
4 ' xgene,ygene,zgene =',3f11.4/
5 ' xmesh,ymesh,zmesh =',3f11.4/
6 ' rotan =',2f10.4/
7 ' 0bcodes =',6i2//
8 ' bvalues =',6f11.4//
9 ' visc =',f11.4/' spqr =',f11.4//
1 1x,80('-')//
1 ' node information'////' nnodes =',i5//
2 ' 8',6x,'1 code side',8x,'x',13x,'y',13x,'z',
3 12x,'phi',10x,'flux'/)
240 format(1x,3i8,5f14.4)
250 format(1x,80('-')////' element information'////' numel = ',i9//
1 7x,'1 node1 node2 transmissivity',7x,'length'/)
260 format(1x,3i9,2q10.6)
      end

```



```

subroutine rstud
implicit real*8 (a-h,o-z)
common/param/ maxele,maxnod,nelene,nnodes,iprint,maxd,igrad,
1          nside,impflx
common/stud/  istud,xstud(20),ystud(20)
character*14 file
character*1  sgrad(2)
data sgrad/'q','l'/

c
c the following lines read data for study regions
c
      open(unit=20,readonly,file='study.dat',status='old',
1      err=270)
      read(20,200) istud,igrad
      igrad=3-igrad
c$dir scalar
      do k = 1,istud
         read(20,290) xstud(k),ystud(k)
c$dir scalar
         do ngrad=1,2
            if(igrad.ne.ngrad)then
               kk=k+10*ngrad*20
               write(file,300) sgrad(ngrad),k
               open (unit=kk,file=file,status='unknown',
1               carriagecontrol='list')
               endif
            enddo
         enddo
270 return
280 format (2i2)
290 format (2f6.2)
300 format ('ellipse',a1,i2.2,'.inp')
end

```

```

subroutine sflux(phi,b,irad,inew)
implicit real*8 (a-h,o-z)

```

```

      version 3.0 (jan 1986)

```

```

common/bcode/  ibcode(6),bvalu(6)
common/coord/  coord(3,1)
common/flux/   flx(1)
common/mesh/   xmesh,ymesh,zmesh,xgene,ygene,zgene,
1             rotan,rotan2,nfrac
common/param/  maxelo,maxnod,nelene,nnodes,1prnt,maxd,lgrad,
1             nside,impflx
common/side/   side(1)
byte sidea
common/stud/   istud,xstud(20),ystud(20)
common/title/title(2)
character*80 title
common/titles/iray,ldate,jobnam,date
character*7 iray,jobnam
character*9 ldate,date
dimension phi(*),b(*),inew(*)
dimension sum(6),dl(6)
character*4 glbloc(2)
data glbloc/'glob','loc'/
data iread/0/

```

```

      if(iread.eq.0) then
        iread=1
        write(11,310) iray,ldate,jobnam,date,title
        write(11,320) xmesh,ymesh
        if (istud.ne.0) then

```

```

c$dir scalar
      do ngrad=1,2
        if(ngrad.ne.lgrad)then

```

```

c$dir scalar
      do n=1,istud
        k=n+10*ngrad*20
        write(k,310) iray,ldate,jobnam,date,title
        write(k,340) glbloc(ngrad),xstud(n),ystud(n)
        write(k,330) xmesh,ymesh
      enddo
    endif
  enddo
endif
endif

```

```

c$dir scalar
do i=1,nside
  sum(1) = 0.d0
  dl(i) = 0.d0
enddo

```

```

      if (nside.eq.2)then
        do m=1,nnodes
          if (side(m).eq.1.or.side(m).eq.7)then
            sum(1) = sum(1) + flx(m)
          elseif (side(m).eq.3)then
            sum(2) = sum(2) + flx(m)
          endif
        enddo
        write(6,420) sum(1),sum(2)
        return
      else
        do m=1,nnodes
          j=side(m)

```

```

        if (j.ne.0) then
            sum(j) = sum(j) + flx(m)
            dl(j) = dl(j) + 1.d0
        endif
    enddo
endif
c
write(6,'(1x)')
if (lprnt.eq.2) write(8,'(1x)')
rot=rotan-180.d0
jrot=1rot*(6-nside)/2
c$dir scalar
do i=1,nside
    j=mod(1-jrot+nside,nside)+1
    rot=rot+90.d0
    flux=abs(sum(j))
    if (flux.ne.0.0d0) then
        vsqrtp = sqrt(1.d0/flux)
        write(6,350) 1,sum(j),vsqrtp
        if (lprnt.eq.2) write(8,350) 1,sum(j),vsqrtp
    else
        vsqrtp=1.d20
        write(6,360) 1,sum(j)
        if (lprnt.eq.2) write(8,360) 1,sum(j)
    endif
enddo
c
c*** write tape 11 for plotting ellipses
c
    if (mod(i,2).eq.0) write(11,370) rot,flux,vsqrtp
enddo
ccc calculate the number of fractures/unit length or area
c intersecting each side
c
    fpul=0.d0
    do i=1,nside
        fpul=fpul+dl(i)
    enddo
    if (nside.eq.4) then
        fpul = (dl(1)+dl(2)+dl(3)+dl(4))/(2*(xmesh+ymesh))
        dl(1) = dl(1)/xmesh
        dl(2) = dl(2)/ymesh
        dl(3) = dl(3)/xmesh
        dl(4) = dl(4)/ymesh
        write(6,380)
        if (1rot.gt.1) then
            disave=dl(1)
c$dir scalar
            do l=1,3
                dl(l)=dl(l+1)
            enddo
            dl(4)=disave
        endif
    else
        fpul = (dl(1)+dl(2)+dl(3)+dl(4)+dl(5)+dl(6))/
1      (2*(xmesh*ymesh+xmesh*zmesh+ymesh*zmesh))
        dl(1) = dl(1)/(ymesh*zmesh)
        dl(2) = dl(2)/(xmesh*zmesh)
        dl(3) = dl(3)/(ymesh*zmesh)
        dl(4) = dl(4)/(xmesh*ymesh)
        dl(5) = dl(5)/(xmesh*zmesh)
        dl(6) = dl(6)/(xmesh*zmesh)
        write(6,390)
    endif
enddo
write(6,400) (lside,dl(lside),lside=1,nside)

```

```

write(6,410) fpu1
c
return
c
310 format(a7,' --- ',a9/a7,' --- ',a9/a80/a80)
320 format(' flow region',22x,' xmesh=',f7.2,' ymesh=',f7.2/)
330 format(' flow region',22x,' xmesh=',f7.2,' ymesh=',f7.2)
340 format(' study region',a4,' a1 gradient',6x,' xstud=',f7.2,
1' ystud=',f7.2)
350 format(' side',i1,' --- the sum of the fluxes =',lpe14.6,
1' --- sqrt(1/sum) =',lpe14.6)
360 format(' side',i1,' --- the sum of the fluxes =',lpe14.6,
1' --- sqrt(1/sum) = infinity')
370 format(3gl5.5)
380 format('/' number of fractures per unit length'/
1' side fractures')
390 format('/' number of fractures per unit area '/'
1' side fractures')
400 format(17,f16.4)
410 format(' global',f14.4/)
420 format(' inner hole ----- the sum of the fluxes =',lpe14.6/
1' outside circle - the sum of the fluxes =',lpe14.6)
end

```

```

subroutine sort1j
implicit real*8 (a-h,o-z)

c
c      version 3.0 (jan 1986)
c      insure that i .lt. j and that i is in increasing order
c
common/trans/ trans(1)
common/dist/ dist(1)
common/inode/ inode(2,1)
common/param/ nmaxe,maxnod,nelene,nnodes,iprnt,maxd,igrad,
1      nside,impflx

c
do i = 1, nelene
c
    n1 = inode(1,i)
    if (n1 .gt. inode(2,i)) then
        inode(1,i) = inode(2,i)
        inode(2,i) = n1
    endif
enddo

c
nmax=nelene
nmin2=2
do while (nmax.ge.nmin2)
    nmax2=nmax
    nfirst=1
    nmin=0
    nmax=0
    do n=nmin2,nmax2
        it=inode(1,n-1)
        if (it.gt.inode(1,n)) then
            inode(1,n-1)=inode(1,n)
            inode(1,n)=it
            it=inode(2,n-1)
            inode(2,n-1)=inode(2,n)
            inode(2,n)=it
            rt=trans(n-1)
            trans(n-1)=trans(n)
            trans(n)=rt
            rt=dist(n-1)
            dist(n-1)=dist(n)
            dist(n)=rt
            nmax=n-1
            nmin=nmin+nfirst*nmax
            nfirst=0
        endif
    enddo
    nmin2=max0(nmin,2)
enddo
return
end

```

```

subroutine study(phi)
implicit real*8 (a-h,o-z)

c
c      version 3.0 (jan 1986)
c handle study regions
c
common/trans/ trans(1)
common/coord/ coord(3,1)
common/inode/ inode(2,1)
common/mesh/  xmesh,ymesh,zmesh,xgene,ygene,zgene,
1  rotan,rotan2,nfrac
common/param/ maxele,maxnod,nelene,nnodes,lprnt,maxd,lgrad,
1  nside,impfix
common/pi/    pi180
common/stud/  istud,xstud(20),ystud(20)
common/vispg/ visc,spqr
dimension phi(1)
dimension nfrac(4,20),flux(4,20),flux1(4)
double precision sumphi(4,20),sumphi2(4,20),avphi(4),ddphi

c
dimension t(4),is(4),s(4)

c
r=rotan*pi180
cosr2=2*cos(r)
sinr2=2*sin(r)

c
c the following 2-d arrays are initialized as 1-d for vectorization
do i = 1, 4*istud
  nfrac(i,1)=0.d0
  flux(i,1)=0.d0
  sumphi(i,1)=0.d0
  sumphi2(i,1)=0.d0
enddo

c
do i=1,nelene
  n1=inode(1,i)
  n2=inode(2,i)
  x1=coord(1,n1)
  y1=coord(2,n1)
  dx=coord(1,n2)-x1
  dy=coord(2,n2)-y1

c
c calculate array t of intersection values
c
nt=0
flen2=dx*dx+dy*dy
denomx= sinr2*dx-cosr2*dy
denomy=-(sinr2*dy+cosr2*dx)
prtx=cosr2*y1-sinr2*x1
prty=cosr2*x1+sinr2*y1
do 420 k=1,istud
  if (denomx.ne.0.) then
    is(1)=1
    is(2)=3
    t(1)=(prtx+xsstud(k))/denomx
    t(2)=(prtx+xsstud(k))/denomx
    nt=2
  else
    if (abs(prty).ge.ystud(k)) go to 420
  endif
  if (denomy.ne.0.) then
    is(nt+1)=2

```

```

        t(nt+2)=(prty-ystud(k))/denomy
        nt=nt+2
    else
        if (abs(prtx).ge.xstud(k)) go to 420
    endif
c
c  sort array t of intersection values
c
        j1=2
        i2=nt
        do while (i2.ge.j1)
            j2=i2
            i0=1
            i1=0
            i2=0
            do j=j1,j2
                if (t(j-1).gt.t(j)) then
                    tt=t(j-1)
                    t(j-1)=t(j)
                    t(j)=tt
                    it=ix(j-1)
                    ix(j-1)=ix(j)
                    ix(j)=it
                    i2=j-1
                    i1=i1+10*i2
                    i0=0
                endif
            enddo
            j1=max0(i1,2)
        enddo
        i1=nt/2
        if (mod(ix(1)+ix(2)+i1,2).eq.0) go to 420
        if (t(i1).ge.1.d0.or.t(i1+1).le.0.d0) go to 420
c
c  element is partially inside study region
c
        nt=0
        if (t(i1).ge.0.d0) then
            ix(1)=ix(i1)
            t(1)=r(i1)
            s(1)=1.d0
            nt=1
        endif
        if (t(i1+1).le.1.) then
            nt=nt+1
            ix(nt)=ix(i1+1)
            t(nt)=t(i1+1)
            s(nt)=-1.d0
        endif
        if (nt.gt.0) then
            phi1=phi(n1)
            dphi=phi1-phi(n2)
            fluxe=trans(i)*dphi/sqrt(fien2)
c$dir scalar
            do i1=1,nt
                iside=ix(i1)
                nfr(i1,side,k)=nfr(iside,k)+1
                flux(iside,k)=flux(iside,k)+s(i1)*fluxe
                ddphi=phi1-t(i1)*dphi
                sumphi(iside,k)=sumphi(iside,k)+ddphi
                sumphi2(iside,k)=sumphi2(iside,k)+ddphi*ddphi
            enddo

```

```

c *** store information about fractures part or all of which fall into
c *** the flow region.
c
c      420      continue
c
c      enddo
c
c$dir scalar
do k=1,istud
write(6,430) k,xstud(k),ystud(k),rotan
nfrct=0
c$dir scalar
do iside=1,4
n=nfrct(iside,k)
nfrct=nfrct+n
if (n.ne.0) then
avphi(iside)=sumphi(iside,k)/n
sdphi=sqrt (sumphi2(iside,k)/n-avphi(iside)**2)
write(6,440) iside,avphi(iside),sdphi
else
write(6,450) iside
endif
enddo
c$dir scalar
do iside=1,3,2
t(iside)=nfrct(iside,k)/xstud(k)
t(iside+1)=nfrct(iside+1,k)/ystud(k)
if (nfrct(2,k).ne.0.and.nfrct(4,k).ne.0)
1 flux1(iside+1)=flux(iside+1,k)*ymesh/
2 (xmesh*(avphi(2)-avphi(4)))
flux(iside,k)=flux(iside,k)*xmesh/xstud(k)
flux(iside+1,k)=flux(iside+1,k)*ymesh/ystud(k)
enddo
fpul=nfrct/(2*(xstud(k)+ystud(k)))
write(6,'(1x)')
c$dir scalar
do iside=1,4
aflux=abs(flux(iside,k))
if (aflux.gt.1.d-20) then
s(iside)=sqrt(1.d0/aflux)
write(6,460) iside,flux(iside,k),s(iside)
else
s(iside)=1.d10
write(6,470) iside,flux(iside,k)
endif
flux(iside,k)=aflux
enddo
write(6,480) (iside,t(iside),iside=1,4),fpul
if (igrad.ne.1) then
kk=k+30
write(kk,490) rotan, flux(2,k),s(2)
write(kk,490) rotan+180.d0,flux(4,k),s(4)
endif
if (igrad.ne.2) then
write(6,500)
if (nfrct(2,k).ne.0.and.nfrct(4,k).ne.0)then
c$dir scalar
do iside=2,4,2
aflux=abs(flux1(iside))
if (aflux.gt.1.d-20) then
s(iside)=sqrt(1.d0/aflux)
write(6,460) iside,flux1(iside),s(iside)
else
s(iside)=1.d10

```



```

        write(6,470) iside,flux1(iside)
    endif
    flux1(iside)=aflux
enddo
else
    write(6,510)
    flux1(2)=flux(2,k)
    flux1(4)=flux(4,k)
endif
    kk=k+50
    write(kk,490) rotan      ,flux1(2),s(2)
    write(kk,490) rotan+180.d0,flux1(4),s(4)
endif
enddo
return

```

c

```

430 format('// study region number',i3/
1 ' the size of study region is',f7.2,' by',f7.2/
2 ' the angle of rotation is',f20.2/)
440 format(' side',i2,'; average head is',lpe14.6,' , standard ',
1 'deviation is',lpe14.6)
450 format(' side',i2,' intersects no fracture')
460 format(' side',i1,' --- the sum of the fluxes =',lpe14.6,
1 ' --- sqrt(1/sum) =',lpe14.6)
470 format(' side',i1,' --- the sum of the fluxes =',lpe14.6,
1 ' --- sqrt(1/sum) = infinity')
480 format(' number of fractures per unit length'/
1 ' side fractures'/
2 4(6x,i1,f16.4/),
3 ' global',f14.4/)
490 format(3gl5.5)
500 format('/' normalized fluxes -- local gradient')
510 format(' average head could not be calculated for side',
1 ' 2 or side 4'/
2 ' because no fracture intersected that side'/
3 ' values written for ellipg are those for global',
4 ' gradient')
end

```

```

subroutine symsol (lbwth,na,a,noddim,b)
implicit real*8 (a-h,o-z)

c
c      version 3.0 (jan 1986)
c
c      matrix solver
c
c      common/param/ maxels,maxnod,nelene,nnodes,iprint,maxd,igrad,
1      nside,implix
dimension b(noddim,2),a(lbwth,na)

c
c      reduce matrix
c
c      do n=1,na
c
c      if a(l,n) is 0, symsol will fail, force soln. by letting a-le-20
c
      if (a(l,n).eq.0.d0) then
        a(l,n) = 1.d-20
        write(6,370) n, a(l,n)
      endif
      an = 1.0d0 / a(l,n)
      li=n
      do l=2,lbwth
        cc=a(l,n)*an
        i = n + l - 1
        if (i.le.na) then
          j = l - 1
          do k=1,lbwth
            a(k-j,i) = a(k-j,i) - cc*a(k,n)
          enddo
        endif
        a(l,n) = cc
      enddo
      enddo

c
c      reduce vector
c
c      c$dir scalar
      do irot = 1,maxd
        do n=1,na
          an = 1.0d0 / a(l,n)
          do l=2,lbwth
            i = n + l - 1
c
c      removed test, although it will perform operations on b(i,*) with
c      i beyond the desired index, the a(*,*) involved should be zero
c      and there will be no net affect
c          if (i.gt.na) go to 360
            b(i,irot) = b(i,irot) - a(l,n)*b(n,irot)
          enddo
c 360      continue
            b(n,irot) = b(n,irot) * an
          enddo

c
c      back substitution
c
      do n = na-1, 1, -1
        do k=2,lbwth
          i = n + k - 1
c
c      removed test, although it will perform operations using b(i,*) with

```

```

c      if (na.ge.1) then
        b(n,1rot) = b(n,1rot) - a(k,n)*b(1,1rot)
c      endif
      enddo
    enddo
  return
370 format (' n=',15,5x,'a(n,1)= ',e10.5)
end

```

```
subroutine werror(line)
character*(*) line
open (unit=10,file='line1.err',status='unknown',
1    carriagecontrol='list')
write(10,520) line
stop
520 format( 'error ',a)
end
```

Appendix G

ELLFMG - Program Organization and Arrays

Table G-1. ELLFMG - Description of Program Variables

angle	Angle between the gradient and the x axis for the directional permeability measurements
ellname	Character variable which contains character string "ELLFMG" - the name of the program. Character *7.
edate	Date on which the program is executed. Character *9.
elix elily e2jx e2jy	Coordinates of eigenvectors
file	Character variable that contains the name of the file to be opened for input: "ELLIPSE.INP", "ELLIPSE01.INP", etc. Character *14.
filel	Character variable that contains the name of the file to be opened for output: plotting information for the program "ELLP". Character *14.
idate	Date of fracture mesh generation by FMG. Character *9.
iray	Label for program output. Character *7.
istud	Maximum number of study regions.
jobname*7	The name of the job read from unit 11. Character *7.
jdate	Date the flow program LINEL was executed. Character *9.
kij	Directional permeability.
maxn	Maximum number of angles.
maxnr	Maximum number of runs.
n	Number of angles per run.
nr	Number of runs.
ngrad	Gradient type ngrad = 0 specified flow region gradient = 1 global gradient = 2 local gradient.
ne	Number of points for plotting the best fit ellipse.

nd	Total number of permeability measurements: number of angles times number of runs.
pkganre	$1/\sqrt{K}$ for one angle
r2	Radius for plotting in polar coordinates.
sum1 - sum7	Different terms in the equation for calculating K11, K12, K22 (see Theory and Design Report, Equations 5-18, 5-19 and 5-20).
theta1	Angle between first eigenvector and x axis.
theta2	Angle between second eigenvector and x axis.
nkganre	The element (i,nr) of the array [xkgan].
xk11 xk12 xk22	Components of the permeability tensor.
xk1	First principal permeability.
xk2	Second principal permeability.
xmse	Mean square error.
xgnmse	Normalized geometric mean square error.
xanmse	Normalized arithmetic mean square error.
xmeank	Mean permeability.

Table G-2. ELLFMG - Description of Program Arrays

an(maxn)	Angle in degrees of a directional permeability as output by LINEL
ella(180)	Angle in degrees for plotting best fit ellipses.
ello(180)	Values of the directional permeability for the best fit ellipse, for plotting.
ellp(180)	$1/\sqrt{r^2}$ for each angle if $r^2 > 0$; 10^{20} if $r^2 \leq 0$.
pkgan(50,25)	One over the square root of the directional permeability as output by LINEL.
pkijnn(50)	One over the square root of the direction permeability for the best fit ellipse.
pkav (50)	One over the square root of the average permeability when several runs are input for each angle.
title(4)*80	Title, input variable read from unit 11.
xkgan(50,25)	Directional permeability as calculated by LINEL.
x1(50,25)	x coordinate of a point in rectangular coordinate system.
x2(50)	x coordinate of a best fit point in rectangular coordinate system.
y1(50,25)	y coordinate of a data point in rectangular coordinate system.
y2(50)	y coordinate of a best fit point in rectangular coordinate system.

Appendix H

ELLFMG - Program Listing

```

program ellimg
c  an = evenly spaced angles from 0 degrees to 360 degrees
c  xkgan = k = permeability as calculated by line1
c  pkgan = 1/(sqrt k )
c  x1 and y1 are all data points in rectangular coor.
c  x2 and y2 are best fit points in rectangular coor.
c  pki_jnn = 1/(kij) or best fit
c  xkav = avg k
c  pkav = 1/(sqrt xkav)
c  n = no. of angles = i
c  nr = no. of runs = j
c
c
c      this program calculates,
c      1) permeability tensors
c      2) principal permeability
c      3) coordinates of eigenvectors
c      4) mean square error
c      5) normalized mean square error
c      6) angle of theta 1 (between e1 and x axis)
c      7) angle of theta 2 (between e2 and x axis)
c      and generates plotting input file.
c
dimension an(50), xkgan(50,25),pkgan(50,25)
character*80 line(20)
dimension xkav (50),pkav(50)
character*80 title(4)
dimension a1(50,25),y1(50,25),x2(50),y2(50),pki_jnn(50)
dimension elip(180),ella(180),ello(180)
character idate*9,iray*7,jobname*7,jdate*9,ellname*7,edate*9
character file*14,file1*14,sgrad(2)*1
integer istud
data sgrad/'g','l'/

c
file='ellipse.inp'
open(unit=6,file='ellimg.out',status='new')
istud=1
do ngrad=0,2
  do nstud=1,istud
    if(ngrad.gt.0) then
20      write(file,20)sgrad(ngrad),nstud
      format('ellipse',a1,i2.2,'.inp')
    endif
    open (unit=11,readonly,file=file,status='old',err=60)
    if(ngrad.gt.0) then
      file1=file
      file1(12:14)='plt'
    else
      file1='ellipse.plt'
    endif
    open (unit=7,file=file1,status='new')

c
    maxn=50
    maxnr=25
    ne = 180
    jlag=0
    iflag=0

c
c      read program i.d.s and dates from img and line1
c      and title as input to img
c
    read(11,240) iray,idate,jobname,jdate,title
    jlag=0
    l2=1

```

```

c      read directional permeability data from line1
c
c      read(11,260)angle,xkqanre,pkqanre
      xkmin = 1.0e20
      absmn= 1.0e-20
      maxx1=manx+1
      do nr = 1,maxnr1
        do i = 1,maxn
          an(i)=angle
          xkqan(i,nr)=xkqanre
          pkqan(i,nr)=pkqanre
          if (xkqan(i,nr) .lt.xkmin) then
            xkmin = xkqan(i,nr)
            anmin = an(i)
          end if
          read(11,260,end=50)angle,xkqanre,pkqanre
          i2=i2+1
          if(angle.eq.an(i))goto 40
        enddo
        write(6,6001)manx
6001      format(' more than',i3,' angles given for one region ')
        stop
        n=i
        enddo
        write(6,6002)manxr
6002      format(' more than',i3,' runs given for one region ')
        stop
        50      if(nr.eq.1)n=i
              nd=n*nr
              if(xkmin.le.absmn)iflag0=1
c
c      check to see if input file contained the correct
c      number of data points
c
      if (i2.ne.nd) then
        write(6,270) nd,i2,file
        stop
      endif
      ellname='ellimg '
      call date (edate)
      write(6,280) ellname,edate,iray,ldate,jobname,jdate
      write(6,290) title
      write(6,300) file,i2,n,nr
c
c      calculate avg kqan values for each angle
c      initialize average
c
      do i = 1,n
        xkav(i) = 0
      enddo
      do i = 1,n
        do j = 1,nr
          xkav(i) = xkav(i) + xkqan(i,j)/nr
        enddo
        write(6,310) an(i),xkav(i)
      enddo
      call sum(an,xkqan,maxn,maxnr,n,nr,sum1,sum2,
1      sum3,sum4,sum5,sum6,sum7,jlag,thetal)
c
c      compute k11,k12,k22 --- permeability tensor
c
      xk22=(sum1*sum5 - sum2*sum3) / (sum4*sum5 - sum2*sum2)
      xk11 = sum3/sum5-sum2*xk22/sum5

```

```

nk12=sum6/sum7
C
C compute lambda 1 and lambda 2 --- principal permeability
C
z=sqrt(((nk11+nk22)**2.-4.*((nk11*nk22)-(nk12)**2))/2
nk1=((nk11+nk22)/2.)+z
nk2=((nk11+nk22)/2.)-z
if (nk2.lt.0) then
write(6,6003)
6003 format(2x,'REGRESSION FAILED')
flag=5
end if

C
C compute coordinates of eigenvectors
C
e1lx=1./sqrt(1.+((nk1-nk11)/nk12)**2.)
e1ly=((nk1-nk11)/nk12)/sqrt(1.+((nk1-nk11)/nk12)**2.)
e2lx=1./sqrt(1.+((nk2-nk11)/nk12)**2.)
e2ly=((nk2-nk11)/nk12)/sqrt(1.+((nk2-nk11)/nk12)**2.)

C
C compute angles between e1 and e2 and x-axis
C
theta1=atan2(e1ly,e1lx)*180/3.14159
theta2=atan2(e2ly,e2lx)*180/3.14159

C
C cccc If k2<0 let k2=kmin,redo regression to find k1
C cccc assuming the eigenvectors are not changed
C
if (flag.eq.5) then
C
C these eqns from V-36 Long phd, insure const. eigenvectors
C
yy1 = (nk1-nk11)/nk12
yy2 = (nk2-nk11)/nk12
C
nk2 = nkmin

C
C call sum again, this time let an = an -theta1 to put
C ccc data into principal coord's
C ccc In this case, K2 is know = kmin
C ccc K12 is zero
C ccc Only need to calculate K1 with eqn.V-22 Long phd
C
call sum(an,nkgn,maxn,maxnr,n,nr,sum1,sum2,
1 sum3,sum4,sum5,sum6,sum7,flag,theta1)
nk1 = (sum3-nk2*sum2)/sum5
nk11 = ((yy1*nk2)-yy2*nk1)/(yy1-yy2)
nk12 = (nk1-nk11)/yy1
nk22 = ((yy1*nk1)-nk12)/yy1
end if

C
C compute coordinates for plotting ellipses in polar
C and rectangular systems
C compute sum8 for mean square error
C compute x1 and y1 for all values
C compute x2 and y2 for best fit ellipse
C
p1180=3.14159/180.
sum8=0.0

```

```

sinan=sin(an1)
cosan=cos(an1)
r2 = xk11*cosan*cosan+2*xk12*sinan*cosan+
      xk22*sinan*sinan
if (r2.gt.0.)then
  pk1jnn(1) = 1.0/sqrt(r2)
else
  pk1jnn(1) = 1.0e20
end if
x2(1) = pk1jnn(1)*cosan
y2(1) = pk1jnn(1)*sinan
if (xhav(1).eq. 0.0) then
  pkav(1)=1.0e20
else
  pkav(1) = 1.0/sqrt(xhav(1))
endif
do j = 1,nr
  sum0=sum0+(xhqan(1,j)-r2)**2
  x1(1,j) = pkqan(1,j)*cosan
  y1(1,j) = pkqan(1,j)*sinan
enddo
enddo

```

compute mean square error and normalized mean square error

```

nmse=sum0/nd
if (11lag0.eq.0)then
  xgmse=nmse/(abs(xk1)*abs(xk2))
else
  xgmse=0.
endif
nmseak=(abs(xk1)+abs(xk2))/2.
xnmse=nmse/nmseak**2
write(6,320) xk1,xk12,xk22,xk1,xk2,elin,elly,
1          e2jx,e2jy,nmse,xgmse,xnmse,theta1,
2          theta2

```

```

write(6,330)
do i=1,n
  write(6,350) an(i)
  do j = 1,nr
    write(6,360)pkqan(1,j),xhqan(1,j)
  enddo
enddo
write(6,360)
do i = 1,n
  write(6,370)an(i),pk1jnn(1),pkav(1)
enddo

```

write data for ellipses in rectangular coordinates

```

write(6,380)
do i=1,n
  write(6,350)an(i)
  do j = 1,nr
    write(6,400)x1(1,j),y1(1,j)
  enddo
enddo
write(6,390)
do i = 1,n
  write(6,370) an(i),x2(i),y2(i)

```

```
prepare data for plotting an ellipse ---
```

```
do i=1,ne
  ella(i)=2.*i
  p180=3.14159/180.
  an1=ella(i)*p180
  sinan=sin(an1)
  cosan=cos(an1)
  r2=xk11*cosan*cosan+2*xk12*sinan*cosan+
  xk22*sinan*sinan
  ello(i)=r2
  if (r2.gt.0.) ellp(i)=1.0/sqrt(r2)
  if (r2.le.0.) ellp(i)=1.0e20
enddo
```

```
write(7,410) icontp, nr
write(7,'(a80)') title
if (jlag.eq.5) then
  write(7,210)
  format(' Regression Failed')
else
  write(7,220)
  format(' Successful Regression')
end if
```

```
write(7,410) nd
write(7,200) jlag
format(' jlag= ',i1)
write(7,430) ((pkqan(i,j),j=1,nr),i=1,n)
write(7,430) ((an(i,j),j=1,nr),i=1,n)
write(7,410) ne
```

```
write(7,430) (ellp(i),i=1,ne)
write(7,430) (ella(i),i=1,ne)
```

```
write(7,430) thetal,sgnase
```

```
write(7,430) ((xkqan(i,j),j=1,nr),i=1,n)
write(7,430) (ello(i),i=1,ne)
close(unit=7)
```

```
enddo
```

```
60 istud=20
```

```
enddo
```

```
stop
```

```
240 format(2(a7,5x,a9/), (a80))
```

```
250 format(2i5)
```

```
260 format(3e15.5)
```

```
270 format('1 The number of data points should be',i4,'.'/
```

```
1 ' There are',i5,' data points on ',a,' -- run terminated.')
```

```
280 format('1 --- ',a7,' --- ',a9/' --- ',a7,' --- ',a9/'
```

```
1 ' --- ',a7,' --- ',a9/')
```

```
290 format('0',a/(1x,a))
```

```
300 format('0The number of data points in ',a,' is',i5,'.'/
```

```
1 ' 1 Permeability was calculated in',i5,' directions.'/
```

```
2 ' 0the number of runs is',i5///
```

```
3 1lx,'an (deg)', ,lx,'kav(cm/sec)')//
```

```
310 format(10x,f9.2,10x,e15.5)
```

```
320 format('1 permeability tensors'//10x,'k11=',e20.10/
```

```
1 10x,'k12=',e20.10/10x,'k22=',e20.10///
```

```
2 ' principal permeability'//10x,'k1 (lambda1)=' ,e20.10/
```

```
3 10x,'k2 (lambda2)=' ,e20.10///' coordinates of eigenvectors'//
```

```
4 14x,'e1ix=' ,e20.10/14x,'e1iy=' ,e20.10/14x,'e2ix=' ,e20.10/
```

```

5 14x,'e2jy-',e20.10///' mean square error-',8x,e20.10//
6 ' geom. normalized mean sq error- ',q20.10///
7 ' arithm. normalized mean sq error-',q20.10///
8 ' angle of theta 1(between e1 and x-axis)',f10.5//
9 ' angle of theta 2(between e2 and x-axis)',f10.5///

c
c 321 format(//10x,'ab data fmg 0013 (64x64)'/10x,'units are cm. sec.'//
c 1 10x,'parameters constant'/10x,'two sets randomly located')
c
330 format('1 parameters for plotting data points in polar',
1 ' coordinates'//
2 10x,'angle alpha',10x,'1/sqrt(kgan)',13x,'xkgan'//)
340 format('1best fit values'//
1 10x,'angle alpha',21x,'1/sqrt(k1jnn)',14x,'1/sqrt(xkav)'//)
350 format('0'/11x,f7.2)
360 format(25x,e15.5,11x,e15.5)
370 format(10x,f7.2,11x,e15.5,12x,e15.5)
380 format('1 parameters for plotting data points in cartesian',
1 ' coordinates'//10x,'angle alpha',13x,' xxy from kgan'//)
390 format('1cartesian coordinates of best fit ellipse'//
1 5x,'angle alpha',13x,'xxy from x1jn'//)
400 format(25x,e15.5)
410 format(1615)
430 format(8e10.4)
end
subroutine sum(ae,xkgan,maxn,maxnr,n,nr,sum1,sum2,sum3,sum4,sum5,
1 sum6,sum7,jlag,theta1)
dimension ae(maxn),xkgan(maxn,maxnr)
pi100=3.14159/100.0
sum1=0.
sum2=0.
sum3=0.
sum4=0.
sum5=0.
sum6=0.
sum7=0.
do 450 i=1,a
an1=ae(i)*pi100
ccc if k2<0, go to principal coord's
c
if(jlag.eq.5)an1 = (ae(i)-theta1)*pi100
c
sinan=sin(an1)
cosan=cos(an1)
sum2=sum2+(cosan*sinan)**2
sum4=sum4+sinan**4
sum5=sum5+cosan**4
sum7=sum7+2*[(sinan*cosan)**2)
do 440 j = 1,nr
sum1=sum1+xkgan(i,j)*sinan**2
sum3=sum3+xkgan(i,j)*cosan**2
sum6=sum6+xkgan(i,j)*cosan*sinan
440 continue
450 continue
xnr=nr
sum1=sum1/xnr
sum3=sum3/xnr
sum6=sum6/xnr
return
end

```

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. Neither the United States Government nor any agency thereof, nor The Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial products process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or The Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or The Regents of the University of California and shall not be used for advertising or product endorsement purposes.

Available to DOE and DOE Contractors from the
Office of Scientific and Technical Information
P.O. Box 62, Oak Ridge, TN 37831
Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, VA 22161
Price: Printed Copy A07, Microfiche A01

Lawrence Berkeley Laboratory is an equal opportunity employer.