Lawrence Berkeley National Laboratory
Recent Work

Title

MODELS FOR CALCULATING FLUID FLOW IN RANDOMLY GENERATED THREE-DIMENSIONAL NETWORKS OF DISC-SHAPED FRACTURES: USERS' MANUALS AND LISTINGS FOR FMG3D, DISCEL, AND DIMES

Permalink

https://escholarship.org/uc/item/46s431z9

Authors

Gilmour, H.M.P.
Billaux, D.
Long, J.C.S.

Publication Date

1986-10-01

eScholarship.org Powered by the California Digital Library
University of California

LBL-19516 c.2

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

EARTH SCIENCES DIVISION

RECEIVED
LAWRENCE
BERKELEY LABORATORY

JAN 30 1987

LIBRARY AND
DOCUMENTS SECTION

MODELS FOR CALCULATING FLUID FLOW IN RANDOMLY GENERATED THREE-DIMENSIONAL NETWORKS OF DISC-SHAPED FRACTURES: USERS' MANUALS AND LISTINGS FOR FMG3D, DISCEL, AND DIMES

H.M.P. Gilmour, D. Billaux, and J.C.S. Long

October 1986

TWO-WEEK LOAN COPY

This is a Library Circulating Copy which may be borrowed for two weeks.



LBL-19516 c.2

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

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, 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 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 product, 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.

LBL-19516

MODELS FOR CALCULATING FLUID FLOW IN RANDOMLY GENERATED THREE-DIMENSIONAL NETWORKS OF DISC-SHAPED FRACTURES

Users' Manuals and Listings for FMG3D, DISCEL, and DIMES

H. M. Peggy Gilmour, Daniel Billaux, and Jane C. S. Long

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

October 1986

This work was supported by the U.S. Department of Energy, Chicago Operations Office for the Crystalline Repository Project. Task No. E504-621AH under Contract No. DE-AC03-76SF00098.

Table of Contents

List of Tables v

1.0 Introduction 1

2.0 Program FMG3D 3

3.0 Program DISCEL 17

4.0 Program DIMES 27

Appendix A: Program FMG3D Organization and Arrays 37

Appendix B: Listing of Program FMG3D 61

Appendix C: Program DISCEL Organization and Arrays 143

Appendix D: Listing of Program DISCEL 157

Appendix E: Program DIMES Organization and Arrays 189

Appendix F: Listing of Program DIMES 199

List of Tables

Table	Title	Page
2-1.	FMG3D Input Variables and Formats	5
2-2.	FMG3D Input Control	7
2-3.	Description of FMG3D Input Variables	8
2-4.	FMG3D Input/Output Files	11
2-5.	Program FMG3D Error Warnings and Stops	12
3-1.	DISCEL Input Variables and Formats	18
3-2.	Description of DISCEL Input Variables	19
3-3.	DISCEL Input/Output Files	20
3-4.	Program DISCEL Error Warnings and Stops	21
4-1.	DIMES Input Variables and Format	28
4-2.	Description of DIMES Input Variables	29
4-3.	DISCEL Input/Output Files	31
A-1.	Description of FMG3D Program Variables	39
A-2.	Description of FMG3D Program Arrays	41
A-3.	FMG3D Subroutine Outline	50
A-4.	Description of FMG3D Subroutines	52
C-1.	Description of DISCEL Program Variables	145
C-2.	Description of DISCEL Program Arrays	149
C-3.	DISCEL Subroutine Outline	152
C-4.	Description of DISCEL Subroutines	153
E-1.	Description of DIMES Program Variables	191
E-2.	Description of DIMES Program Arrays	194
E-3.	DIMES Subroutine Outline	196
E-4.	Description of DIMES Subroutines	197

1.0 INTRODUCTION

The purpose of this report is to provide the user with sufficient information to run the three programs FDG3D, DISCEL and DIMES. A previous report explained the theory and design of the programs (Models for Calculating Fluid Flow in Randomly Generated Three-Dimensional Networks of Disc-Shaped Fractures). 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, and program warning and error stops. In addition, information not strictly needed to run the programs, but useful in understanding the internal structure and flow is provided in appendices. The appendices cover program variables and arrays, subroutines outlines, a short description of each subroutine, and finally listings of the codes. The additional information on FMG3D, DISCEL, and DIMES is in Appendices A, C and E respectively, and the listings are in Appendices B, D, and F.

The codes are written in FORTRAN-77 for use on VAX computers. Large arrays are dimensioned by constants set in "parameter" statements. An "include" statement is used to insert an overall maximum dimensioning parameter statement in each main program and appropriate subroutine. For each program, the allowable problem size, controlled by the maximum number of fractures, nodes, etc., can be changed simply by editing the included parameter statement. Common blocks are also inserted in the code by include statements. Preceding each FORTRAN listing is a list of the files, maximum dimensions and common blocks, that are included by the programs.

2.0 PROGRAM FMG3D

Several options are available for output generated by program FMG3D. First the user selects one of three modes of operation, or ways in which the program can be run:

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

The mode of operation is controlled by the input variable icont, (the values of icont, 1, 2, and 3, correspond to the option numbers above). The user then controls the generation 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 2-2 shows which input groups are read for each value of control variable icont. Except for Group 2, all input variables are read from the file FMG3D.INP; Groups 1 and 7 are read by the main program, Groups 3, 4, 5, and 6 are read by subroutine FRAGEN. Group 2, the primary fracture system, is read from file FRAC3D.DAT, and is unformatted. This data group is written if icont equals 1 or 2, but contains the fracture system from a previous run when icont equals 3. Data Group 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 sphere, and of various orientations. The nodal segmentation parameters nseg and nsegb, may be specified but if the line is left blank, they are set to default values of 1 and 3, respectively. Data Group 2 is read for each flow region when icont equals 2 or 3 to restore the primary fracture system which is altered when fractures are truncated or deleted. Input variables are described in Table 2-3.

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

Table 2-5 lists the various error messages and error stops which may occur when running FMG3D. The most likely would be exceeding maximum dimensions.

Three sample input files, corresponding to the three modes of program operation, are given following Table 2-5. Variable labels are included in input files for the convenience of the user; they are not read by the program.

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

Table 2-1. FMG3D Input Variables and Formats

Group	Variable	Format	Input Unit
1	icont,iplot,imesh iplane,idump,iord	10x,15,2(15x,i5)	1
2*	iray,idate		5
	title(i),i=1,40		
	nsets,itole		
	iseti(i,j),j=1,8,i=1,nsets		
	rseti(i,j),j=1,16,i=1,nsets		
	nfrac,rgene		
	(iold,(frac(j,iold),j=1,12),isetfr(iold),		
	nflag(iold),icubk(iold)) iold=iref(i),i=1,nfrac		
	ncub,ilengc		
	ifcuk(i),i=1,ncub		
	icub(i),ifcu(i),i=1,ilengc		
3	iranf,dseed	10x,i5,15x,d15.8	1
	title(i),i=1,20	20a4	
	title(i),i=21,40	20a4	
	rgene	10x,f10.4	
	nsets,itole	2(10x,i5)	
4	icent,idens	2(10x,i5)	1
5(icent=1)			
	nfrac	10x,i5	1
	frac(i,j),i=1,7,j=1,nfrac	7f10.4	
6(icent=2)			
	nfrac (if dens=1)	10x,i5	1
	or		
	rlamb (if dens=0)	10x,f10.4	
	ichar (for phi and theta)	10x,i5	
	const (if ichar=2)	10x,f10.4	
	const (if ichar=2)	10x,f10.4	
	or		
	idist,ev,sd,ycept (if ichar=3)	10x,i5,3(5x,f10.4)	
	idist,ev,sd,ycept (if char=3)	10x,i5,3(5x,f10.4)	
	ichar (for radius)	10x,i5	
	const (if ichar=2)	10x,f10.4	
	or		
	idist,ev,sd,ycept (if ichar=3)	10x,i5,3(5x,f10.4)	

	ichar (for aperture)	10x,i5	
	const (if ichar=2)	10x,f10.4	
	or		
	idist,ev,sd,ycept (if ichar=3)	10x,i5,3(5x,f10.4)	
7	xmesh,ymesh,zmesh	3(10x,f10.4)	1
	rophi,rothe	2(10x,f10.4)	
	nseg,nsegb	10x,i5,15x,i5	

* This group, which was written by a previous generation, is read from an unformatted VAX file FRAC3D.DAT when icont=3.

Table 2-2. FMG3D Input Control

Program Mode	Value of icont	Input Groups*
generation only	1	1, 3, 4, 5 or 6 (repeat 4 and either 5 or 6 for each fracture set)
generation and mesh	2	1, 3, 4, 5, or 6, 7 (repeat 4 and either 5 or 6 for each fracture set; repeat 7 for each mesh)
mesh only, use previous generation	3	1, 2, 7 (repeat 7 for each mesh)

*see Table 2-1 for input groups

Table 2-3. Description of FMG3D Input Variables

Variable	Description
const	Constant value for orientation angles, radius or aperture
dseed	Seed for random number generator, type real*8
ev	Expected value of statistical distribution
icent	Code for determining fracture characteristics, by set = 1 - read orientation, radius, aperture and coordinates of fracture center = 2 - statistically generate coordinates of fracture center; either set orientation angles, radius, and aperture to constant values or statistically generate them
ichar	Code for determining individual fracture characteristics; ichar is read for each of orientation, radius and aperture = 2 - read constant value = 3 - statistically generate values
icont	Code for controlling program execution = 1 - generate primary fracture system only = 2 - generate primary fracture system and mesh = 3 - read primary fracture system from local file and generate mesh
idens	Code for input of number of fractures per set or fracture density per set; one is input and the other computed = 0 - input rlamb (density), compute nfrac = 1 - input nfrac, compute rlamb
idist	Code for distribution to be used for generating values = 1 - normal distribution = 2 - lognormal distribution = 3 - exponential distribution = 4 - normal distribution correlating aperture with radius = 5 - uniform distribution over specified range (other distribution options may be added to fit field data)
idump	Code for printing of a "dump" file FMG3D.DMP: fracture and intersection information after each change in the fracture mesh. = 0 no dump file written = 1 a dump file is written
imesh	Code for producing mesh for fluid flow model = 0 - no mesh

	= 1 - generate mesh
iord	Code for dividing into subregions = 0 or 1 no division > 1 divide into iord × iord × iord cubes
iplane	Code for printing input file for program PLANE = 0 no printing = 1 file PLANE.INP is printed set iplane to 0 (program PLANE doesn't exist yet)
iplot	Code for producing input files for plot programs = 0 - no plot files = 1 - generate plot files
iranf	Code for duplicating random number generation of previous run = 0 - random number generation = 1 - read dseed to duplicate previous run
itole	Number of decimal places in coordinates of fracture center
nfrac	Number of fractures per set
nseg	Number of segments into which each internal node is divided, default - nseg = 1
nsegb	Number of segments into which each boundary node is divided, default - nsegb = 3
nsets	Number of fracture sets
rgene	Radius of generation sphere
rlamb	Number of fractures per unit volume (density)
rophi	Second angle of rotation of flow region
rothe	First angle of rotation of flow region
sd	Standard deviation of statistical distribution, or half range for uniform distribution (idist = 5)
title(40)	Two lines of 80 character titles for labeling output
xmesh	Width of flow region in x-direction
ycept	Parameter for normal distribution correlating one characteristic with another, idist=4

ymesh	Width of flow region in y-direction
zmesh	Width of flow region in z-direction

Table 2-4. FMG3D Input/Output Files

Unit	Name	Read/ Write	Main Prog./ Subroutine	Description
1	FMG3D.INP	Read	FMG3D	Input data Groups 1 and 7*
			FRAGEN	Input data Groups 3, 4, 5,and 6*
2	FMGPR.DAT	Write		Program data for print out
			FMG3D	Title page, program options, error messages, etc.
			PFS	Statistics on characteristics of fracture sets for fracture systems
			STAT2D	Statistics on length of boundary intersections, per set, for same fracture systems as above
4	FMG3D.DMP	Write	WFRAC1	Fracture characteristics by sets of: all generated fractures, truncated fractures in generation region, truncated fractures in flow region
			WINTER	All boundary nodes; boundary nodes and fracture intersections of conducting fractures only
5	FRAC3D.DAT	Write	WRFIVE	All primary fracture system data: file is written when icont = 1 or 2, then read in for each flow region in the run, or in subsequent runs when icont = 3, input data Group 2*
		Read	WRFIVE	
6	DISCEL.DAT	Write	WNODES	Input data for fluid flow model: all pertinent run identification, flow region, fracture system, and nodal arrays and variables
7	PLANE.INP		FMG3D	Title and controls
			WINTER	mesh information
8	DIMES01.DAT DIMES02.DAT		WDIMES	Input data for plotting program

* see Table 2-1 for input groups

Table 2-5. Program FMG3D Error Warnings and Stops

Program or Subroutine	Message Format Number	Warning or Stop	Reason and Action, if any
FMG3D	385	Warning	Input flow region exceeds generation region limit. Go to next flow region.
	390	Warning	All fractures are outside the flow region. Go to next flow region.
	395	Warning	There are no conducting fractures in the flow region. Go to next flow region.
AREAT	110	Warning	Numerical problems when looking for disc - line intersections skip fracture and go on.
	120	Warning	
BNDINT	110	Stop	Numerical problems: boundary lines in fracture plane appear to intersect at more than twelve points.
	120	Stop	From call to COMSEG, logical test failed.
	130	Stop	From call to BNDSEG's call to COMSEG, logical test failed.
BNDNOD	120	Stop	From call to COMSEG, logical test failed.
	130	Stop	Maximum dimension maxint exceeded.
BNDSEG	110	Stop	From call to COMSEG, logical test failed.
CIRCPT	100	Stop	Trying to define a line between the center of a fracture and a point, the point is at the center of the fracture.
COMSEG	110		Sets error flag for calling program if logical test for common segment fails due to numerical problems.
DISCOUT	10	Stop	Maximum dimension maxout*3 exceeded.
FRAGEN	System	Stop	Incomplete data set, reading End of File.
	300	Stop	Maximum dimension maxfrc exceeded.

	310	Stop	Maximum dimension maxfrc exceeded.
INTERS	150	Stop	Maximum dimension maxint exceeded.
INTFRC	150	Stop	From call to COMSEG, logical test failed.
NODELE	110	Stop	Maximum dimension maxnod exceeded.
STATOR	2010	Warning	Due to the spread in fracture orientations, the statistical calculations on orientation for a given set may be off.
TRUNCF	110	Stop	From call to AREAT's call to CIRCPT, numerical problems: less than two roots from call to SIMULC when there must be two roots.
TRUNCN	130	Stop	From call to SIMULL, numerical problems: perpendicular lines appear to be parallel.
	135	Stop	Numerical problems: fracture intersection appears to be truncated at more than two boundary nodes.
WPLOT	180	Stop	Maximum dimension maxnpf exceeded when iopt = 1.
	190	Stop	Maximum dimension maxnpf exceeded when iopt = 2.

FMG3D Sample Input File, icont = 1

```
ICONT- - -    1     IPLOT- - -    0     IMESH- - -    0
iplane - -    1     IDUMP- - -    1     IORD - - -    5
IRANF- - -    1     RSEED- - -50311543.d0
This is an example of network plotting using DIMES
August 25, 1986
rgene- - -   1.
NSETS- - -    3  ITOLE- -    2
ICENT- - -    2  IDENS --    1
RLAMBNFRAC  200
ICHAR- - -    3
idist- - -    1         90.            25.
idist- - -    1          0.            25.
ICHAR- - -    3
idist- - -    1          0.3            0.
ichar- - -    3
idist- - -    2          0.001          0.
ICENT- - -    2  IDENS- -    1
RLAMBNFRAC  300
ICHAR- - -    3
idist- - -    1          0.            25.
idist- - -    1          0.            20.
ichar- - -    3
idist- - -    1          0.2            0.
ICHAR- - -    3
idist- - -    2          0.001          0.
ICENT- - -    1  IDENS- -    1
RLAMBFRAC     2
       5.0     270.0      0.50     0.001     -0.55      0.30     0.100
      80.0       5.0      0.30     0.001     -0.60     -0.10     0.000
```

FMG3D Sample Input File, icont = 2

```
ICONT- - -    2     IPLOT- - -    1     IMESH- - -    1
IPLANE - -    0     IDUMP- - -    1     IORD - - -    2
IRANF- - -    1     RSEED- - -500311543.d0
This is a test of DISCEL using a small random
system.
rgene- - -50.
NSETS- - -    1  ITOLE- -    2
icent- - -    1
nfrac- - -   10
   45.0000  225.0000   16.0000    0.0010    6.0000    5.0000    7.0000
   20.0000  170.0000    8.0000    0.0010   -3.0000   -1.0000    4.0000
   10.0000  190.0000    7.0000    0.0010    1.0000    2.0000    3.0000
  130.0000   10.0000   12.0000    0.0010    5.0000    3.0000   -5.0000
  140.0000  200.0000   15.0000    0.0010   -6.0000    2.0000   -4.0000
  110.0000   45.0000   10.0000    0.0010    8.0000   -6.0000   -2.0000
   30.0000  170.0000    9.0000    0.0010   -5.0000    4.0000    6.0000
   20.0000   20.0000    8.0000    0.0010    4.0000   -7.0000    8.0000
   45.0000   60.0000    6.0000    0.0010   -3.0000   -2.0000   -5.0000
   80.0000  110.0000    7.0000    0.0010    2.0000   -5.0000    3.0000
xmesh- - -   20.0000  ymesh- -   20.0000  zmesh- -   20.0000
rophi- - -    0.0000  rothe- -    0.0000
nseg - - -    1     nsegb- - -    1
```

This input deck corresponds to the small random case described and plotted in section 5.2.2 of the report on theory and design of FMG3, DISCEL, and DIMES

FMG3D Sample Input File, icont = 3

```
ICONT- - -    3      IPLOT- - -    0      IMESH- - -    1
iplane - -    0      IDUMP- - -    0      IORD - - -    0
xmesh- - -    5.       ymesh- -    2.       zmesh- -    0.5
rophi- - -    0.       rothe- -    0.
nseg - - -    1      nsegb- - -    3
xmesh- - -    5.       ymesh- -    2.       zmesh- -    0.5
rophi- - -   30.       rothe- -   50.
nseg - - -    1      nsegb- - -    3
```

3.0 PROGRAM DISCEL

The fluid flow program computes flux through each of the fracture networks output by FMG3D for boundary conditions read in DISCEL. The user may input any number of boundary conditions per fracture mesh; the program can solve for up to three at a time. Only constant head and no flow boundaries can be modeled at this time.

Table 3-1 lists the program input variables and their formats by groups. Table 3-2 describes the input variables.

Table 3-3 lists the Input/Output units used by the program and tells what is read or written by DISCEL and its subroutines.

Table 3-4 presents a list of error warnings and stops that may occur in operating the code. The most likely would again be exceeding maximum dimensions.

Following Table 3-4 is a sample input file for running DISCEL with a description of what would happen if this set of inputs were used. This input data would be read from file DISCEL.INP (unit 1). A small sample fracture mesh generated by FMG3D and written on file DISCEL.DAT follows. Further information about the program is given in Appendix C: description of the program variables and arrays (Tables C-1 and C-2), a subroutine outline (Table C-3) and descriptions of the program subroutines (Table C-4). Appendix D is a listing of the code.

Table 3-1. DISCEL Input Variables and Formats

Group	Variable	Format	Input Unit
1	iunits,ninteg	2(10x,i5)	1
2	newbc	10x,i5	1
3	ibcc	10x,i5	1
	ndir(3)	10x,3i5	
	delphi(3)	10x,3f10.4	
	or		
	seth(6,maxd)	10x,6f10.4	
4*	iray,idate	a7,5x,a9	2
	ititle(40)	20a4	
	xmesh,ymesh,zmesh	3(10x,f10.4)	
	rophi,rothe	2(10x,f10.4)	
	(label)	10x	
	frcoef(6,4)	4f10.4	
	nfrac	10x,i5	
	(label)	10x	
	frac1(nfrac,8),coefp(nfrac,4)	7f20.4,5e12.4	
	nnode	10x,i5	
	nseg,nsegb	2(10x,i5)	
	(label)	10x	
	node1(nnode,6)	6f10.4	
	(label)	10x	
	nside(nnode,2)	5(5x,2i5)	
	(label)	10x	
	node2(nnode,2)	5(5x,2i5)	
	node3(1)	10x,i5	
	node3(i),i=2,node3(1)	15i5	
	(label)	10x	
	nkey(nfrac,2)	5(5x,2i5)	

*Group 4 is written by program FMG3D and read from the local VAX file DISCEL.DAT.

Table 3-2. Description of DISCEL Input Variables

Variable	Description
delphi(3)	Gradient delta phi, if ibcc = 1; up to three values corresponding to flow directions ndir(3)
ibcc	Boundary condition type: = 1 constant gradient = 2 set head for each boundary plane
iunits	Type of units to be used: = 1 c g s = 2 m k s
ndir(3)	Flow directions, up to three: = 1 inflow through face 1, outflow through face 3; = 2 inflow through face 5, outflow through face 6; = 3 inflow through face 2, outflow through face 4
newbc	The number of times new boundary conditions, Group 3, will be read for the next fracture mesh, Group 4: > 1 read Group 3 for next mesh and complete all calculations,then read another Group 3 and complete calculations using the same mesh and the new set(s) of boundary conditions, repeat for newbc groups of boundary conditions; = 1 read one group of boundary conditions for next mesh; = 0 use previous boundary conditions for next mesh: = EOF use previous boundary conditions for next mesh and all subsequent meshes. Note, newbc must be > 0 for the first mesh
ninteg	The number of integration steps to be used when approximating shape functions.
seth(6,maxd)	Set head values for the six flow region boundary planes, if ibcc = 2; up to three sets of values corresponding to flow directions ndir(3).
Group 4.	Fracture network data from FMG3D; described in Tables 2-3, A-1, and A-2.

Table 3-3. DISCEL Input/Output Files

Unit	Name	Read/ Write	Main Prog./ Subroutine	Description
1	DISCEL.INP	Read	DISCEL	Input data Groups 1 and 2*
			BNDCON	Input data Group 3
2	DISCEL.DAT	Read	READ2	Input data Group 4 from FMG3D
3	PRINT.OUT	Write	DISCEL	Title page, run identification, program parameters, error messages, etc.
			BNDCON	Boundary conditions and error messages
			ROFRAC	Parameters for each fracture: fracture number, radius, aperture, and transmissivity; parameters for each node on the current fracture: global and local node numbers, node length, half the width, and node type.
			SFLUX	Flux and average head for each node in each of the mdir flow directions for both the individual fractures and for the entire fracture network. Total flux through each of the six boundary planes in the mdir flow directions. (Note, fluxes are not computed for nodes on fractures that do not intersect boundary planes, therefore, some printed flux values may be meaningless.)

* see Table 3-1 for input groups

Table 3-4. Program DISCEL Error Warnings and Stops

Program or Subroutine	Message Format Number	Warning or Stop	Reason and Action, if any
DISCEL	190	Stop	Normal stop at end of fracture mesh data, ndata = 2 from READ2.
	191 192	Stop	Incomplete fracture mesh data set from FMG3D, ndata = 3 from READ2.
	193 194	Stop	Maximum dimension maxfrc or maxnod exceeded.
	193 198	Stop	Maximum dimension maxn exceeded.
BNDCON	140 135	Stop	Incomplete boundary condition input, reading EOF.
	140 145	Stop	Boundary condition option ibcc out of range.
	140 150	Stop	No flow directions specified, i.e., a blank line.
	140 170	Stop	Maximum dimension maxh exceeded.
CGSOLV	100	Stop	Iterative matrix solver did not converge to a solution
FORMG	100	Stop	Matrix inversion error, ier = 129 or ier = 131 from call to LINV2F.

DISCEL Sample Input File; DISCEL.INP

```
iunits - -    1  ninteg -   10
newbc- - -    3
ibcc - - - .  1
directions    3    1    2
delta phis    1.0000    1.0000   -1.0000
ibcc - - -    2
directions    1    2
set heads1    1.0000    0.5000    0.0000    0.5000    1.0000    0.0000
set heads2    0.0000    0.0000    0.0000    0.0000    5.0000    2.0000
ibcc - - -    1
directions    3    3
delta phis    5.0000   3.5000
newbc- - -    0
newbc- - -    1
ibcc - - -    1
directions    1    2    3
delta phis    1.0000    1.0000    1.0000
EOF
```

If this input file were used:

DISCEL would be run using cgs units and ten integration steps to approximate shape functions.

For the first mesh output by FMG3D, boundary conditions, Group 3, would be read three times. Flux would be computed first for constant gradient boundary conditions in three flow directions with gradients of 1.0, 1.0, and -1.0. The first flow direction is from side 2 to side 4 of the flow region, the second from side 1 to side 3, and the third from side 6 to side 5 because $\Delta\phi$ is negative. Next, using the same fracture mesh, flux would be computed in two directions with the heads set for each of the six boundary planes. Finally, flux would be computed for constant gradient boundary conditions with flow from side 5 to side 6, for two values of $\Delta\phi$, 5.0 and 3.5.

For the second fracture mesh output by FMG3D, flux would be computed using the previous Group 3 data, i.e.:

ibcc = 1, ndir(1) = 3, ndir(2) = 3, maxd = 2, and

delphi(1) = 5.0, and delphi(2) = 3.5.

For the third fracture mesh, flow would be computed using constant gradient boundary conditions, in maxd = 3 flow directions, which are ndir(i) = 1, 2, 3 and delphi(i) = 1.0, 1.0, 1.0, for i = 1,maxd.

All subsequent meshes would use the last set of boundary conditions, since an EOF would be encountered. This assumes there are three or more fracture meshes on input unit 2. If there were less than three, an EOF would be encountered by READ2 and the run stopped before all the data on input unit 1 could be read.

The minimum data required to run DISCEL, therefore, is one each of input Groups 1, 2, 3, and 4, with newbc = 1, one flow direction, one gradient, and one fracture mesh.

Sample FMG3D Output File for Input to DISCEL.DAT

```
fmg3d   --- 29-AUG-86
this is a test of DISCEL using a small random
system.
xmesh- - -   20.0000ymesh- - -   20.0000zmesh- - -   20.0000
rophi- - -    0.0000rothe- - -    0.0000
     frcoef
     1.0000    0.0000    0.0000  -10.0000
     0.0000    0.0000    1.0000  -10.0000
     1.0000    0.0000    0.0000   10.0000
     0.0000    0.0000    1.0000   10.0000
     0.0000    1.0000    0.0000  -10.0000
     0.0000    1.0000    0.0000   10.0000
nfrac- - -   10
     frac
    45.0000  225.0000   16.0000    0.0010    6.0000    5.0000    7.0000  0.3084E+03 -0.5000E+00  0.5000E+00  0.7071E+00 -0.4456E+01
    20.0000  170.0000    8.0000    0.0010   -3.0000   -1.0000    4.0000  0.1988E+03 -0.3368E+00 -0.5939E-01  0.9397E+00 -0.4829E+01
    50.0000  190.0000   12.0000    0.0010    5.0000    3.0000   -5.0000  0.2656E+03 -0.7544E+00  0.1330E+00  0.6428E+00  0.6587E+01
    40.0000   20.0000   15.0000    0.0010   -6.0000    2.0000   -4.0000  0.2841E+03  0.6040E+00 -0.2198E+00  0.7660E+00  0.7128E+01
    70.0000  225.0000   10.0000    0.0010    8.0000   -6.0000   -2.0000  0.1440E+03 -0.6645E+00  0.6645E+00  0.3420E+00  0.9987E+01
    30.0000  170.0000    9.0000    0.0010   -5.0000    4.0000    6.0000  0.1908E+03 -0.4924E+00 -0.8682E-01  0.8660E+00 -0.7311E+01
    20.0000   20.0000    8.0000    0.0010    4.0000   -7.0000    8.0000  0.1213E+03  0.3214E+00 -0.1170E+00  0.9397E+00 -0.9622E+01
    10.0000  190.0000    7.0000    0.0010    1.0000    2.0000    3.0000  0.1539E+03 -0.1710E+00  0.3015E-01  0.9848E+00 -0.2844E+01
    80.0000  110.0000    7.0000    0.0010    2.0000   -5.0000    3.0000  0.1539E+03 -0.3368E+00 -0.9254E+00  0.1736E+00 -0.4474E+01
    45.0000   60.0000    6.0000    0.0010   -3.0000   -2.0000   -5.0000  0.1131E+03  0.3536E+00 -0.6124E+00  0.7071E+00  0.3371E+01
nnode- - -   38
     nseg      1     nsegb      1
     node1
    10.0000   10.0000    6.2929   10.0000    4.7574   10.0000
    10.0000    4.7574   10.0000   -2.7874   -8.0300   10.0000
    10.0000   10.0000    6.2929   -4.5171   10.0000   -3.9723
     3.4130    3.0434    6.5542   -7.6185   -1.9498    2.2845
     4.0997    8.2667    3.3464   -5.7449    0.5035    1.8746
    -3.7823   10.0000   -3.4527   -7.0208    5.1278   -2.2975
     0.6200   -2.5553    8.5382   -6.9967   -3.8622    4.0765
     4.5086    0.9421    8.8147   -1.8993   -7.1419   10.0000
     1.4656   -3.5254    9.8220   -4.1644   -2.3758    5.0281
   -10.0000   -3.7907    1.3145  -10.0000    2.1066    1.6873
    -4.5486   -2.5514    3.3469    3.5994   -4.9980    6.1128
    10.0000   10.0000   -0.5804   10.0000   -4.9155    2.5063
     1.9741   10.0000  -10.0000   -0.8803   -6.1881  -10.0000
    10.0000   10.0000   -0.5804    1.9741   10.0000  -10.0000
     2.9976   10.0000   -8.7987   -0.4194   -3.5743  -10.0000
     4.7887   -8.7960   -2.8069    7.3786   -7.6484   -0.0047
     0.0198   -5.0759   -9.1737    2.5838   -1.0766   -6.9922
     4.9373   -7.1884   -2.9652    7.5208   -7.5469    0.1410
   -10.0000   10.0000    1.4499  -10.0000  -10.0000   -4.2899
     4.5212   10.0000  -10.0000   -2.7582  -10.0000  -10.0000
   -10.0000   10.0000    1.4499    4.5212   10.0000  -10.0000
   -10.0000  -10.0000   -4.2899   -2.7582  -10.0000  -10.0000
    -8.6091   -3.5461   -3.5344   -2.7450   -6.4678   -8.9967
    10.0000   -0.0092   -9.7531   10.0000   -8.8286    7.3808
     9.6506   -0.2315  -10.0000    2.2315   -7.6506  -10.0000
     8.7037  -10.0000    7.1381    0.9720  -10.0000   -7.8827
     8.4083   -7.5550    1.8140    6.6879   -7.5557   -1.5269
    10.0000   -8.0253    5.8202    8.2133  -10.0000    6.1855
     1.1070    9.2635   10.0000    2.5389    1.1426   10.0000
   -10.0000   10.0000    3.7586  -10.0000   -2.8113    2.4943
   -10.0000   10.0000    3.7586    0.5553   10.0000    9.7802
     1.9892    0.5697    9.6300    1.9608   -0.5002    9.5066
    -3.6809   -2.3481    6.1136   -0.1212   -3.2815    8.0440
```

```
  10.0000  -10.0000    5.5744   10.0000   -1.9019    6.5825
  -2.9395  -10.0000   10.0000    0.7037    0.0096   10.0000
  10.0000  -10.0000    5.5744   -2.9395  -10.0000   10.0000
   7.7119   -6.3641    6.8096    0.5994   -3.2461    9.6304
  -3.8503   -3.0000    2.3109    1.9798   -4.9207    3.3819
nside
        1     1          2     2          5     5          0     0          0     0
        5     0          0     0          0     2          0     0          3     3
        0     0          1     1          4     4          5     5          5     4
        0     0          0     0          0     0          3     3          4     4
        5     5          6     6          0     0          1     1          4     4
        6     6          0     0          1     6          2     2          3     3
        5     5          0     0          0     0          1     1          2     2
        6     6          0     0          0     0
node2
        1    -1          1    -2          1    -5          1     2          1     8
        1     4          1     6          1     7          1     9          2    -3
        2     9          3    -1          3    -4          3    -5          3     4
        3     5          3    10          3     9          4    -3          4    -4
        4    -5          4    -6          4    10          5    -1          5    -4
        5    -6          5     9          5     7          6    -2          6    -3
        6    -5          6     7          6     9          7    -1          7    -2
        7    -6          7     9          8     9
node3       57 entries
    1     2     3     4     5     6     7     8     9    10     4    11    12    13    14
   15    16    17    18    19    20    21    22     6    15    23    24    25    26    16
   27    28    29    30    31     7    32    33    34    35    36     8    28    32    37
    5    38     9    11    18    27    33    37    38    17    23
nkey
        9     2          3    11          7    14          7    21          6    28
        6    34          7    40          2    47          7    49          2    58
```

4.0 PROGRAM DIMES

The plotting program draws plots of the disc network produced by FMG3D using plot specifications read in DIMES. The user may chose to run the program interactively or to submit it as a batch job. If DIMES is run interactively, the plot specifications are asked for on screen and entered from the keyboard. If DIMES is run in batch mode, the plot specifications are read from file DIMES.INP. In both cases DIMES processes the files DIMES01.DAT, DIMES02.DAT, etc. from FMG3D in sequence. Table 4-1 lists the program input variables and their formats by group. Table 4-2 describes the input variables. Table 4-3 lists the input/output units used by the program and tells what is read or written by DIMES and its subroutines. Following Table 4-3 is a sample input file for running DIMES in batch. This input data would be read from file DIMES.INP, data from FMG3D being read from file DIMESnn.DAT, where nn is the number of a flow region. Further information about the program is given in Appendix E: A description of the program variables and arrays (Tables E-1 and E-2) a subroutine outline (Table E-3), and descriptions of the program subroutines (Table E-4). Appendix F is a listing of the code.

Table 4-1. DIMES Input Variables and Format

Group	Variable	Format	Input Unit
1	ititle1(i), i=1,20	20a4	1
	ititle2(i), i=1,20	20a4	
2	xgene, ygene, zgene	3f10.4	1
	xmesh, ymesh, zmesh	3f10.4	
	rophi, rothe	2f10.4	
	nfrac, nint, nont	3i10	
	iord	i10	
	apmax,radmax	2f10.4	
	numfrc(i), i=1, ncub	10i6	
	numinc(i), i=1, ncub	10i6	
	ifrok(i), i=1, ncub+1	10i6	
3	*Chap	a4	2
	(if chap = "view"or "VIEW")		
	viewphi,viethe,npol	2f10.4,i10	
	(if chap = "cube" or "CUBE"		
	ix1, ix2, iy1, iy2, iz1, iz2	6i10	
	(if chap = "flow" or "FLOW")		
	open a new data file and		
	read new flow region data		
	(back to group 1)		
4**	ic	i6	1
	i,frac(k,i),k=1,7,*numfrc(ic)times*	i6,7e10.3	
	i,end(k,i),k=1,3,5,2,4,6,ifint(k,i),k=7,8 *(ifrok(ic+1)-ifrok(ic)+numinc(ic))times*	i6,6f10.4,2i10	

*Group 3 is read this way only in batch mode. If the program is run in interactive mode, the variables viewphi, viewthe, npol, ix1, ix2, iy1, iy2, iz1, iz2 are asked for, and entered at the keyboard in free format. After completion of a plot, the user is then asked

- do you want to change the cubes in the subregion ("yes" equivalent to chap. = "CUBE")
- do you want to change the view point ("yes" equivalent to chap. = "VIEW")
- do you want to change the flow region ("yes" equivalent to chap. = "FLOW")

**read for each cube selected in the subregion to be plotted.

Table 4-2. Description of DIMES Input Variables

Variable	Description
end(6,maxseg)	Endpoints of intersection and later of any segment in the plot i=1, nout+nint end(1,i) = x_1 end(2,i) = x_2 end(3,i) = y_1 end(4,i) = y_2 end(5,i) = z_1 end(6,i) = z_2
frac(7,maxfrc)	fracture characteristics i=1, nfrac frac(1,i) = phi, frac(2,i) = theta } orientation angles frac(3,i) = radius frac(4,i) = aperture frac(5,i), frac(6,i), frac(7,i) } x, y, z coordinates of fracture center
ifint(2,maxseg):	Identification numbers of fractures making up intersection, or later any segment i = 1,nout+nint ifint(7,i) = first fracture ifint(8,i) = fracture (positive), boundary plane (negative -1 to -6) or nothing (0)
ititle1(20) ititle2(20)	Equivalent to title in FMG3D; two lines of 80 character titles for labeling output.
ix1	Lowest number of row (in x direction) of cubes to be incorporated in the plotting subregion ix1 = 1 to iord.
ix2	Highest number of row (in x direction) of cubes to be incorporated in the plotting subregion ix2 = ix1 to iord.
iy1	Lowest number of column (in y direction) of cubes to be incorporated in the plotting subregion iy1 = 1 to iord.
iy2	Highest number of column (in y direction) of cubes to be incorporated in the plotting subregion iy2 = iy1 to iord.
iz1	Lowest number of pillar (in z direction) of cubes to be incorporated in the plotting subregion iz1 = 1 to iord.
iz2	Highest number of pillar (in z direction) of cubes to be incorporated in the plotting subregion iz2 = iz1 to iord.
npol	Number of segments making up the broken line representing

	each disk.
viewphi	View angle ϕ_v, as described in the report on programs theory and design, Section 4-2, Figure 4-1.
viewthe	View angle θ_v, as described in the report on programs theory and design, Section 4-2, Figure 4-1.
Group 2	Variables from FMG3D are described in Tables 2-3, A-1, and A-2.

Table 4-3. DISCEL Input/Output Files

Unit	Name	Read/ Write	Main Prog./ Subroutine	Description
1	DIMESnn.DAT	Read	RDATA	Input data Groups 1, 2, 4
2	DIMES.INP (batchmode)	Read	RDATA	Input data Group 3
5	(Keyboard-interactive mode)	read	DIMES	Code for new viewpoint, subregion or flow region
			RDATA	Input data Group 3
6	(screen-interactive mode)	write	DIMES	promt for code for new viewpoint, subregion, or flow region
			RDATA	prompt for Group 3 data
			DRAW(DISPLA)	drawing of a plot
	(name dependent on device)	write	DRAW(DISPLA)	plot file for specified plotting device

Sample Input File for DIMES: DIMES.INP

If this input file is used, DIMES makes two plots. Both have the same view point specification: $\phi_v = 20°$, $\theta_v = 10°$, npol = 30. One shows the fractures intersecting a 3 cubes $\times$ 3 cubes $\times$ 3 cubes subregion. The other one shows the fractures intersecting one cube. These two plots are then repeated for any number of flow regions output by FMG3D.

```
view
        20.       10.       30
cube
          1         3         1         3         1         3
cube
          2         2         2         2         2         2
```

Sample Output File from FMG3D to be Input

in DIMES: DIMES01.DAT

```
this is a test of DISCEL using a small random
system.
  100.0000  100.0000  100.0000
   20.0000   20.0000   20.0000
    0.0000    0.0000
        10        58         7
         2
 1.000E-03 1.600E+01
     5     6     4     5     8     9     7     8
    26    30    21    26    35    38    34    38
     1     6    13    18    25    32    39    46    53
     1
     1 4.500E+01 2.250E+02 1.600E+01 1.000E-03 6.000E+00 5.000E+00 7.000E+00
     3 5.000E+01 1.900E+02 1.200E+01 1.000E-03 5.000E+00 3.000E+00-5.000E+00
     4 4.000E+01 2.000E+01 1.500E+01 1.000E-03-6.000E+00 2.000E+00-4.000E+00
    10 4.500E+01 6.000E+01 6.000E+00 1.000E-03-3.000E+00-2.000E+00-5.000E+00
     9 8.000E+01 1.100E+02 7.000E+00 1.000E-03 2.000E+00-5.000E+00 3.000E+00
     1   -1.4103   13.5687   -4.2989   -3.7823   10.0000   -3.4527          1          4
     2   -2.6500   -8.0891   10.1389   -1.8993   -7.1419   10.0000          1          7
     3   -1.0237   -5.9747  -10.2124   -0.4194   -3.5743  -10.0000          3          4
     4    3.6382   12.5446   -8.5735    2.9976   10.0000   -8.7987          3          4
     5  -12.6964    2.7677    1.5004  -11.8860    8.6575    2.5517          4          6
     1   10.0000   10.0000    6.2929   10.0000    4.7574   10.0000          1         -1
     2   10.0000    4.7574   10.0000   -2.7874   -8.0300   10.0000          1         -2
     3   10.0000   10.0000    6.2929   -4.5171   10.0000   -3.9723          1         -5
    21    3.4130    3.0434    6.5542   -7.6185   -1.9498    2.2845          1          2
    22    4.0997    8.2667    3.3464   -5.7449    0.5035    1.8746          1          8
    23   -3.7823   10.0000   -3.4527   -7.0208    5.1278   -2.2975          1          4
    24    0.6200   -2.5553    8.5382   -6.9967   -3.8622    4.0765          1          6
    25    4.5086    0.9421    8.8147   -1.8993   -7.1419   10.0000          1          7
    26    1.4656   -3.5254    9.8220   -4.1644   -2.3758    5.0281          1          9
     5   10.0000   10.0000   -0.5804   10.0000   -4.9155    2.5063          3         -1
     6    1.9741   10.0000  -10.0000   -0.8803   -6.1881  -10.0000          3         -4
     7   10.0000   10.0000   -0.5804    1.9741   10.0000  -10.0000          3         -5
    28    2.9976   10.0000   -8.7987   -0.4194   -3.5743  -10.0000          3          4
    29    4.7887   -8.7960   -2.8069    7.3786   -7.6484   -0.0047          3          5
    30    0.0198   -5.0759   -9.1737    2.5838   -1.0766   -6.9922          3         10
    31    4.9373   -7.1884   -2.9652    7.5208   -7.5459    0.1410          3          9
     8  -10.0000   10.0000    1.4499  -10.0000  -10.0000   -4.2899          4         -3
     9    4.5212   10.0000  -10.0000   -2.7582  -10.0000  -10.0000          4         -4
    10  -10.0000   10.0000    1.4499    4.5212   10.0000  -10.0000          4         -5
    11  -10.0000  -10.0000   -4.2899   -2.7582  -10.0000  -10.0000          4         -6
    32   -8.6091   -3.5461   -3.5344   -2.7450   -8.4678   -8.9967          4         10
    27   -4.5486   -2.5514    3.3469    3.5994   -4.9980    6.1128          2          9
    33    8.4083   -7.5550    1.8140    6.6879   -7.5557   -1.5269          5          9
    36   -3.6809   -2.3481    6.1136   -0.1212   -3.2815    8.0440          6          9
    37    7.7119   -6.3641    6.8096    0.5994   -3.2461    9.6304          7          9
    38   -3.8503   -3.0000    2.3109    1.9790   -4.9207    3.3819          8          9
     2
     1 4.500E+01 2.250E+02 1.600E+01 1.000E-03 6.000E+00 5.000E+00 7.000E+00
     3 5.000E+01 1.900E+02 1.200E+01 1.000E-03 5.000E+00 3.000E+00-5.000E+00
     4 4.000E+01 2.000E+01 1.500E+01 1.000E-03-6.000E+00 2.000E+00-4.000E+00
     5 7.000E+01 2.250E+02 1.000E+01 1.000E-03 8.000E+00-6.000E+00-2.000E+00
    10 4.500E+01 6.000E+01 6.000E+00 1.000E-03-3.000E+00-2.000E+00-5.000E+00
     9 8.000E+01 1.100E+02 7.000E+00 1.000E-03 2.000E+00-5.000E+00 3.000E+00
     1   -1.4103   13.5687   -4.2989   -3.7823   10.0000   -3.4527          1          4
     2   -2.6500   -8.0891   10.1389   -1.8993   -7.1419   10.0000          1          7
     3   -1.0237   -5.9747  -10.2124   -0.4194   -3.5743  -10.0000          3          4
     4    3.6382   12.5446   -8.5735    2.9976   10.0000   -8.7987          3          4
     5  -12.6964    2.7677    1.5004  -11.8860    8.6575    2.5517          4          6
```

```
 6   11.5665   -6.2940    5.5000   10.0000   -8.0253    5.8202          5          7
 7    7.0379  -11.2991    6.4258    8.2133  -10.0000    6.1855          5          7
 1   10.0000   10.0000    6.2929   10.0000    4.7574   10.0000          1         -1
 2   10.0000    4.7574   10.0000   -2.7874   -8.0300   10.0000          1         -2
 3   10.0000   10.0000    6.2929   -4.5171   10.0000   -3.9723          1         -5
21    3.4130    3.0434    6.5542   -7.6185   -1.9498    2.2845          1          2
22    4.0997    8.2667    3.3464   -5.7449    0.5035    1.8746          1          8
23   -3.7823   10.0000   -3.4527   -7.0208    5.1278   -2.2975          1          4
24    0.6200   -2.5553    8.5382   -6.9967   -3.8622    4.0765          1          6
25    4.5086    0.9421    8.8147   -1.8993   -7.1419   10.0000          1          7
26    1.4656   -3.5254    9.8220   -4.1644   -2.3758    5.0281          1          9
 5   10.0000   10.0000   -0.5804   10.0000   -4.9155    2.5063          3         -1
 6    1.9741   10.0000  -10.0000   -0.8803   -6.1881  -10.0000          3         -4
 7   10.0000   10.0000   -0.5804    1.9741   10.0000  -10.0000          3         -5
28    2.9976   10.0000   -8.7987   -0.4194   -3.5743  -10.0000          3          4
29    4.7887   -8.7960   -2.8069    7.3786   -7.6484   -0.0047          3          5
30    0.0198   -5.0759   -9.1737    2.5838   -1.0766   -6.9922          3         10
31    4.9373   -7.1884   -2.9652    7.5208   -7.5459    0.1410          3          9
 8  -10.0000   10.0000    1.4499  -10.0000  -10.0000   -4.2899          4         -3
 9    4.5212   10.0000  -10.0000   -2.7582  -10.0000  -10.0000          4         -4
10  -10.0000   10.0000    1.4499    4.5212   10.0000  -10.0000          4         -5
11  -10.0000  -10.0000   -4.2899   -2.7582  -10.0000  -10.0000          4         -6
32   -8.6091   -3.5461   -3.5344   -2.7450   -6.4678   -8.9967          4         10
12   10.0000   -0.0092   -9.7531   10.0000   -8.8286    7.3808          5         -1
13    9.6506   -0.2315  -10.0000    2.2315   -7.6506  -10.0000          5         -4
14    8.7037  -10.0000    7.1381    0.9720  -10.0000   -7.8827          5         -6
33    8.4083   -7.5550    1.8140    6.6879   -7.5557   -1.5269          5          9
34   10.0000   -8.0253    5.8202    8.2133  -10.0000    6.1855          5          7
27   -4.5486   -2.5514    3.3469    3.5994   -4.9980    6.1128          2          9
36   -3.6809   -2.3481    6.1136   -0.1212   -3.2815    8.0440          6          9
37    7.7119   -6.3641    6.8096    0.5994   -3.2461    9.6304          7          9
38   -3.8503   -3.0000    2.3109    1.9790   -4.9207    3.3819          8          9
 3
 1 4.500E+01 2.250E+02 1.600E+01 1.000E-03 6.000E+00 5.000E+00 7.000E+00
 3 5.000E+01 1.900E+02 1.200E+01 1.000E-03 5.000E+00 3.000E+00-5.000E+00
 4 4.000E+01 2.000E+01 1.500E+01 1.000E-03-6.000E+00 2.000E+00-4.000E+00
10 4.500E+01-6.000E+01 6.000E+00 1.000E-03-3.000E+00-2.000E+00-5.000E+00
 1   -1.4103   13.5687   -4.2989   -3.7823   10.0000   -3.4527          1          4
 2   -2.6500   -8.0891   10.1389   -1.8993   -7.1419   10.0000          1          7
 3   -1.0237   -5.9747  -10.2124   -0.4194   -3.5743  -10.0000          3          4
 4    3.6382   12.5446   -8.5735    2.9976   10.0000   -8.7987          3          4
 5  -12.6964    2.7677    1.5004  -11.8860    8.6575    2.5517          4          6
 1   10.0000   10.0000    6.2929   10.0000    4.7574   10.0000          1         -1
 2   10.0000    4.7574   10.0000   -2.7874   -8.0300   10.0000          1         -2
 3   10.0000   10.0000    6.2929   -4.5171   10.0000   -3.9723          1         -5
21    3.4130    3.0434    6.5542   -7.6185   -1.9498    2.2845          1          2
22    4.0997    8.2667    3.3464   -5.7449    0.5035    1.8746          1          8
23   -3.7823   10.0000   -3.4527   -7.0208    5.1278   -2.2975          1          4
24    0.6200   -2.5553    8.5382   -6.9967   -3.8622    4.0765          1          6
25    4.5086    0.9421    8.8147   -1.8993   -7.1419   10.0000          1          7
26    1.4656   -3.5254    9.8220   -4.1644   -2.3758    5.0281          1          9
 5   10.0000   10.0000   -0.5804   10.0000   -4.9155    2.5063          3         -1
 6    1.9741   10.0000  -10.0000   -0.8803   -6.1881  -10.0000          3         -4
 7   10.0000   10.0000   -0.5804    1.9741   10.0000  -10.0000          3         -5
28    2.9976   10.0000   -8.7987   -0.4194   -3.5743  -10.0000          3          4
29    4.7887   -8.7960   -2.8069    7.3786   -7.6484   -0.0047          3          5
30    0.0198   -5.0759   -9.1737    2.5838   -1.0766   -6.9922          3         10
31    4.9373   -7.1884   -2.9652    7.5208   -7.5459    0.1410          3          9
 8  -10.0000   10.0000    1.4499  -10.0000  -10.0000   -4.2899          4         -3
 9    4.5212   10.0000  -10.0000   -2.7582  -10.0000  -10.0000          4         -4
```

```
10  -10.0000   10.0000    1.4499    4.5212   10.0000  -10.0000         4          -5
11  -10.0000  -10.0000   -4.2899   -2.7582  -10.0000  -10.0000         4          -6
32   -8.6091   -3.5461   -3.5344   -2.7450   -8.4678   -8.9967         4          10
 4
 1 4.500E+01 2.250E+02 1.600E+01 1.000E-03 6.000E+00 5.000E+00 7.000E+00
 3 5.000E+01 1.900E+02 1.200E+01 1.000E-03 5.000E+00 3.000E+00-5.000E+00
 4 4.000E+01 2.000E+01 1.500E+01 1.000E-03-6.000E+00 2.000E+00-4.000E+00
 5 7.000E+01 2.250E+02 1.000E+01 1.000E-03 8.000E+00-6.000E+00-2.000E+00
10 4.500E+01 6.000E+01 6.000E+00 1.000E-03-3.000E+00-2.000E+00-5.000E+00
 1   -1.4103   13.5687   -4.2989   -3.7823   10.0000   -3.4527         1          4
 2   -2.6500   -8.0891   10.1389   -1.8993   -7.1419   10.0000         1          7
 3   -1.0237   -5.9747  -10.2124   -0.4194   -3.5743  -10.0000         3          4
 4    3.6382   12.5446   -8.5735    2.9976   10.0000   -8.7987         3          4
 5  -12.6964    2.7677    1.5004  -11.8860    8.6575    2.5517         4          6
 6   11.5665   -6.2940    5.5000   10.0000   -8.0253    5.8202         5          7
 7    7.0379  -11.2991    6.4258    8.2133  -10.0000    6.1855         5          7
 1   10.0000   10.0000    6.2929   10.0000    4.7574   10.0000         1         -1
 2   10.0000    4.7574   10.0000   -2.7874   -8.0300   10.0000         1         -2
 3   10.0000   10.0000    6.2929   -4.5171   10.0000   -3.9723         1         -5
21    3.4130    3.0434    6.5542   -7.6185   -1.9498    2.2845         1          2
22    4.0997    8.2667    3.3464   -5.7449    0.5035    1.8746         1          8
23   -3.7823   10.0000   -3.4527   -7.0208    5.1278   -2.2975         1          4
24    0.6200   -2.5553    8.5382   -6.9967   -3.8622    4.0765         1          6
25    4.5086    0.9421    8.8147   -1.8993   -7.1419   10.0000         1          7
26    1.4656   -3.5254    9.8220   -4.1644   -2.3758    5.0281         1          9
 5   10.0000   10.0000   -0.5804   10.0000   -4.9155    2.5063         3         -1
 6    1.9741   10.0000  -10.0000   -0.8803   -6.1881  -10.0000         3         -4
 7   10.0000   10.0000   -0.5804    1.9741   10.0000  -10.0000         3         -5
28    2.9976   10.0000   -8.7987   -0.4194   -3.5743  -10.0000         3          4
29    4.7887   -8.7960   -2.8069    7.3786   -7.6484   -0.0047         3          5
30    0.0198   -5.0759   -9.1737    2.5838   -1.0766   -6.9922         3         10
31    4.9373   -7.1884   -2.9652    7.5208   -7.5459    0.1410         3          9
 8  -10.0000   10.0000    1.4499  -10.0000  -10.0000   -4.2899         4         -3
 9    4.5212   10.0000  -10.0000   -2.7582  -10.0000  -10.0000         4         -4
10  -10.0000   10.0000    1.4499    4.5212   10.0000  -10.0000         4         -5
11  -10.0000  -10.0000   -4.2899   -2.7582  -10.0000  -10.0000         4         -6
32   -8.6091   -3.5461   -3.5344   -2.7450   -8.4678   -8.9967         4         10
12   10.0000   -0.0092   -9.7531   10.0000   -8.8286    7.3808         5         -1
13    9.6506   -0.2315  -10.0000    2.2315   -7.6506  -10.0000         5         -4
14    8.7037  -10.0000    7.1381    0.9720  -10.0000   -7.8827         5         -6
33    8.4083   -7.5550    1.8140    6.6879   -7.5557   -1.5269         5          9
34   10.0000   -8.0253    5.8202    8.2133  -10.0000    6.1855         5          7
 5
 1 4.500E+01 2.250E+02 1.600E+01 1.000E-03 6.000E+00 5.000E+00 7.000E+00
 2 2.000E+01 1.700E+02 8.000E+00 1.000E-03-3.000E+00-1.000E+00 4.000E+00
 8 1.000E+01 1.900E+02 7.000E+00 1.000E-03 1.000E+00 2.000E+00 3.000E+00
 3 5.000E+01 1.900E+02 1.200E+01 1.000E-03 5.000E+00 3.000E+00-5.000E+00
 4 4.000E+01 2.000E+01 1.500E+01 1.000E-03-6.000E+00 2.000E+00-4.000E+00
 6 3.000E+01 1.700E+02 9.000E+00 1.000E-03-5.000E+00 4.000E+00 6.000E+00
 7 2.000E+01 2.000E+01 8.000E+00 1.000E-03 4.000E+00-7.000E+00 8.000E+00
 9 8.000E+01 1.100E+02 7.000E+00 1.000E-03 2.000E+00-5.000E+00 3.000E+00
 1   -1.4103   13.5687   -4.2989   -3.7823   10.0000   -3.4527         1          4
 2   -2.6500   -8.0891   10.1389   -1.8993   -7.1419   10.0000         1          7
 3   -1.0237   -5.9747  -10.2124   -0.4194   -3.5743  -10.0000         3          4
 4    3.6382   12.5446   -8.5735    2.9976   10.0000   -8.7987         3          4
 5  -12.6964    2.7677    1.5004  -11.8860    8.6575    2.5517         4          6
 6   11.5665   -6.2940    5.5000   10.0000   -8.0253    5.8202         5          7
 7    7.0379  -11.2991    6.4258    8.2133  -10.0000    6.1855         5          7
 1   10.0000   10.0000    6.2929   10.0000    4.7574   10.0000         1         -1
 2   10.0000    4.7574   10.0000   -2.7874   -8.0300   10.0000         1         -2
```

APPENDIX A

Program FMG3D Organization and Arrays

Table A-1. Description of FMG3D Program Variables

Variable	Description	How Value is Assigned
Global:		
idate	Current date, character*9	Set by calling VAX FORTRAN subroutine DATE
ilengc	Number of entries in arrays icub and ifcu	calculated in SUBINT
inst	Starting location, minus one, of intersections in arrays intel, inte2, and inside	Calculated in INTFRA
iray	Label for output, character*7	Set in main program
maxfrc	Maximum number of fractures allowed, used for dimensioning arrays	Set in parameter statement in main program
maxint	Maximum number of intersections, used for dimensioning arrays	Set in parameter statement in main program
maxnod	Maximum number of boundary nodes, used for dimensioning arrays	Set in parameter statement in main program
maxord	Maximum value for the subregion division parameter iord, used for dimensioning arrays	Set in parameter statement in main program
maxout	Maximum number of intersections outside the flow region, used for dimensioning arrays	Set in parameter statement in main program
maxset	Maximum number of fracture sets, used for dimensioning arrays	Set in parameter statement in main program
ncub	Number of cubes (or subregions) ncub = iord**3	Calculated in SUBINT
ndiscd	Number of nonconducting fractures discarded	Calculated in INTFRA
nflow	Number of the flow region currently studied	Incremented in main program
nfrac	Total number of fractures in fracture system	Calculated in FRAGEN, adjusted in DISCAR
nfracb	Number of fractures intersecting a flow region boundary	Calculated in BNDNOD
ninb	Number of boundary nodes	Calculated in BNDNOD

nint	Number of fracture intersections	Calculated in INTERS
nnode	Number of nodes	Calculated in NODELE
nout	Number of intersections outside the flow region.	Calculated in TRUNCN, adjusted in DISCOUT.
Local:		
dk	x-coordinate of center of fracture being tested in local coordinate system	Calculated in ROFRAC
flarea	Truncated area of fracture nf in	Calculated in AREAT
ierr	Error condition flag	Set in AREAT, CIRCPT, and COMSEG
iopt	Option flag for subroutine calls	
kib	Counter for total number of boundary nodes	Calculated in BNDNOD
nbn	Number of boundary nodes on fracture nf	Calculated in BNDNOD
nbp	Number of points of intersection between boundary lines in the fracture plane, 4, 8, or 12.	Calculated in BNDINT
ifr, ifrac	Identification number of the fracture being tested	Index of fracture loops in TRUNCF, INTERS, etc. (=1,nfrac)
nroots	Number of roots in simultaneous solution of equations of a given line and a given circle	Calculated in SIMULC
npara	Flag indicating parallel lines	Set of SIMULL
rf	Radius of fracture being tested	Set to frac1(nf,3) in TRUNCF, INTERS, etc.
zplane	Plane of fracture being tested, zplane = -d	Calculated in ROFRAC

Table A-2. Description of FMG3D Program Arrays

Variable	Description
Global:	
binte(6,maxint)	Coordinates of boundary nodes endpoints, calculated in subroutine BNDNOD, k=1,ninb, equivalence (binte,intel) binte(1,k) = x1 binte(2,k) = y1 binte(3,k) = z1 binte(4,k) = x2 binte(5,k) = y2 binte(6,k) = z2 Coordinates are in local fracture system until fracture intersections are truncated in subroutine INTERS, then subroutine RFRCGLO is called to rotate them to the global XYZ coordinate system.
endout(6,maxout)	Coordinates of endpoints for fracture intersections outside the flow region, k = 1, nout, set in subroutine TRUNCF in local coordinates, then rotated to global coordinates by subroutine RFRCGLO endout(1,k) = x1 endout(2,k) = y1 endout(3,k) = z1 endout(4,k) = x2 endout(5,k) = y2 endout(6,k) = z2
frac(12,maxfrc)	Fracture characteristics, read, set or generated in subroutine FRAGEN, n=1,nfrac frac(1,n) = phi, frac(2,n) = theta } orientation angles frac(3,n) = radius frac(4,n) = aperture frac(5,n) = xc, frac(6,n) = yc, frac(7,n) = zc } coordinates of fracture center frac(8,n) = area Coefficients of fracture plane equations, ax + by + cz + d = 0, calculated in subroutine EQPLAN, frac(9,n) = a = lambda frac(10,n) = b = mu frac(11,n) = c = nu frac(12,n) = d = -perpendicular distance from origin to plane
frcoef(6,4)	Coefficients of flow region boundary plane equations

in the global XYZ coordinate system, calculated in subroutine ROFLOW, i=1,6
frcoef(i,1)= ai [lambda 1, mu 1, or nu 1]
frcoef(i,2)= bi [lambda 2, mu 2, or nu 2]
frcoef(i,3)= ci [lambda 3, mu 3, or nu 3]
frcoef(i,4)= di [e.g., -xmesh/2]

ibinte(2,maxint) — Fracture and side that intersect to form boundary node, set in subroutine BNDNOD, k=1,ninb, equilvance (ibinte,inte2)
ibinte(1,k) = identification number of fracture
ibinte(2,k) = - boundary side number

ibkey(maxfrac,2) — Key to boundary nodes on each fracture, set in subroutine BNDNOD, n=1,nfrac
ibkey(n,1) = the number of boundary nodes on fracture n
ibkey(n,2) = the starting position of these nodes in binte, ibinte, and ibside
i.e., entries ibkey(n,2) to ibkey(n,2)+ibkey(n,1)-1

ibside(2,maxint) — Side codes for endpoints of boundary nodes, set in subroutine BNDNOD, equivalence (ibside, inside), k=1,ninb
ibside(1,k) = side code for endpoint 1
ibside(2,k) = side code for endpoint 2
i.e., number of boundary plane on which node lies

icub(maxfrc*2) — Cube/fracture catalog identifying which cubes are intersected by each fracture, set in subroutine SUBINT, j=1,ilengc
entries in this array are keyed by icubk(nfrac)
icub(j) = cube identification number

icubk(maxfrc) — Key to cubes intersected by each fracture, set in subroutine SUBINT, n = 1,nfrac
icubk(n) = starting position of cubes intersected by fracture n in catalog array icub

ifcu(maxfrc*2) — Fracture/cube catalog identifying which fractures intersect each cube, set in subroutine SUBINT, j = 1, ilengc
entries in this array are keyed by ifcuk(ncub)
ifcu(j) = fracture identification number

ifcuk(maxcub) — Key to fractures intersecting each cube, set in subroutine SUBINT, ic = 1,ncub
ifcuk(ic) = starting position of fractures intersecting cube ic in catalog array ifcu

ifout(maxout,2) — Identification numbers of the two fractures involved in an intersection outside the flow region, set in subroutine TRUNCF, k = 1,nout
ifout(k,1) = one of the fractures

	ifout(k,2) = the other fracture
ifro(3*maxout)	Catalog array of intersections outside flow region identifying which intersections are part of a fracture intersecting each cube, set in subroutine DISCOUT; j = 1, ifrok(ncub+1) ifro(j) = intersection identification number
ifrok(maxcub+1)	Key to intersections outside flow region part of fractures in each cube, set in subroutine DISCOUT, n=1, ncub+1 ifrok(n) = starting position of identification numbers of intersections on fractures in cube n in catalog array ifro
inew(-6:maxcfrc)	Back reference array for fracture numbers, set in subroutine SUBINT, modified in subroutines BNDNOD and DISCAR ifold=1, nfrac, where nfrac is the number of fractures generated inew(ifold) = ifrac, such that iref(ifrac) = ifold inew(i) = i if i = -6 to -1
inkey(maxfrc,2)	Key to intersections on each fracture, set in subroutine INTERS, n=1,nfrac, inkey(n,1) = number of intersections on fracture n inkey(n,2) = starting position of identification numbers of intersections on fracture n in catalog array inte3 i.e., entries inkey(n,2)to inkey(n,2)+inkey(n,1)-1
inside(2,maxint)	Truncation side codes for endpoints of fracture intersections, set in subroutine TRUNCN, k=inst+1,inst+nint, equivalence (inside,ibside) inside(1,k) = side code for endpoint 1 inside(2,k) = side code for endpoint 2 i.e., number of the boundary plane on which endpoint lies, or zero if endpoint is internal
intel(6,maxint)	Coordinates of fracture intersection endpoints, k=inst+1, inst+nint, real, equivalence(intel,binte) intel(1,k) = x1 intel(2,k) = y1 intel(3,k) = z1 intel(4,k) = x2 intel(5,k) = y2 intel(6,k) = z2 calculated in subroutine INTERS in local fracture coordinate system, then rotated by subroutine RFRCGLO to the global XYZ coordinate system
inte2(2,maxint)	Identification numbers of the two fractures involved in the intersection, set in subroutine INTERS, k=inst+1,inst+nint, equivalence (inte2,ibinte) inte2(1,k) = one of the fractures inte2(2,k) = the other fracture

inte3(maxint*2+1) Intersection/fracture catalog identifying which intersections are on each fracture, set in subroutine INTERS, j=2,inte3(1):
entries in this array are keyed by inkey(nfrac,2);
inte3(1) = number of entries in the array
inte3(j) = intersection identification number

iref(maxfrc) Reference array for fracture numbers, set in subroutine SUBINT, modified in subroutines BNDNOD and DISCAR, ifrac=1, nfrac
iref(ifrac) = old number ifold of fracture numbers ifrac, when it was generated. The characteristics of fracture ifrac, its intersections, are found in location ifold in the appropriate array.

isetfr(maxfrc) Number of the set in which the fracture was generated, set in FRAGEN
n=1,nfrac

iseti(maxset,8) Code for origin of fracture characteristics, read or set in subroutine FRAGEN,
i=1,nsets, all elements are not necessarily filled
iseti(i,1) = icent: =1 -- read; =2 -- generate
iseti(i,2) = nfrac -- number of fractures per set
iseti(i,3) = ichar (orientation)
iseti(i,4) = idist (orientation)
iseti(i,5) = ichar (radius)
iseti(i,6) = idist (radius)
iseti(i,7) = ichar (aperture)
iseti(i,8) = idist (aperture)
with ichar = 1 -- read values
ichar = 2 -- set to constant
ichar = 3 -- generate according to idist
and idist = 1 -- normal distribution
idist = 2 -- lognormal distribution
idist = 3 -- exponential distribution
idist = 4 -- normal distribution correlating aperture with radius
idist = 5 -- uniform distribution

iside(3,2) Correlates unrotated flow region sides with their normal axes, x, y, and z, and correlates parallel boundary planes, set in data statement in main program
sides 1 and 3 normal to the x-axis
sides 5 and 6 normal to the y-axis
sides 2 and 4 normal to the z-axis

nflag(maxfrc) Contains code to discard, save, or truncate fracture n, set in subroutine FLAG, BNDINT, BNDNOD, INTFRA or TRUNCG, n=1,nfrac

$$\text{nflag(n)} \begin{cases} < 0 \text{ discard} \\ = 0 \text{ save, no truncation} \\ > 0 \text{ truncate at flow region boundary} \end{cases}$$

nkey(maxfrc,2) Key to nodes on each fracture, set in subroutine NODELE, n=1,nfrac
nkey(1,n) = number of nodes on fracture n
nkey(2,n) = starting position of identification numbers of nodes on fracture n in catalog array node3
i.e., entries nkey(2,n) to nkey(2,n)+nkey(1,n)-1 in array node3

node1(maxnod,6) Coordinates of endpoints of nodes, compiled in subroutine NODELE, nn=1,nnode, real
node1(nn,1) = x1
node1(nn,2) = y1
node1(nn,3) = z1
node1(nn,4) = x2
node1(nn,5) = y2
node1(nn,6) = z2

node2(maxnod,2) Identification numbers of two fractures or fracture and side that intersect to form the node, set in subroutine NODELE, nn=1,nnode
node2(nn,1) = fracture identification number
node2(nn,2) = fracture number or -side number

node3(maxnod*2+1) Node/fracture catalog identifying which nodes are on each fracture, compiled in subroutine NODELE, nj=2,node3(1); entries in this array are keyed by nkey(nfrac,2);
node3(1) = number of entries in this array
node3(nj) = node identification number

nside(maxnod,2) Truncation side codes for endpoints of nodes, set in subroutine NODELE, nn=1,nnode
nside(nn,1) = side code for endpoint 1
nside(nn,2) = side code for endpoint 2
i.e., number of boundary plane on which endpoint lies, or zero if endpoint is internal

rseti(maxset,16) Contains either constant values or parameters for chosen distribution for each characteristic, read in subroutine FRAGEN, i=1,nsets
rseti(i,1) = const. phi
rseti(i,2) = expected value of phi
rseti(i,3) = stand. deviat. of phi
rseti(i,4) = const. theta
rseti(i,5) = expected value of theta
rseti(i,6) = stand. deviat. of theta
rseti(i,7) = const. radius
rseti(i,8) = expected value of radius
rseti(i,9) = stand. deviat. of radius
rseti(i,10) = const. aperture/ycept (if idist=4)
rseti(i,11) = exp. value of aperture/slope (if idist=4)
rseti(i,12) = stand. deviat. of aperture

rseti(i,13) = fracture density
rseti(i,14) = const. area (computer from const. radius)
rseti(i,15) = exp. value of area } (from original values
rseti(i,16) = stan. dev. of area } computed from radii)

Local

axy(6,4) — Boundary node endpoints for the 6 sides to be used by subroutine AREAT, values calculated in subroutine BNDNOD, i=1,6
axy(i,1) = x1
axy(i,2) = y1
axy(i,3) = x2
axy(i,4) = y2

bline(12,2) — Contains coordinates, in local fracture system, of up to 12 points of intersection of the 6 boundary lines in the fracture plane, entries are identified by array keybl, calculated in subroutine BNDINT

bxy(4,2) — Used by subroutine AREAT to store the four points involved in computing the area to be trucated on side i, i=1,6
bxy(1,1) = axy(i,1) = x1
bxy(1,2) = axy(i,2) = y1
bxy(2,1) = axy(i,3) = x1
bxy(2,2) = axy(i,4) = y2
bxy(3,1) = x3
bxy(3,2) = y3
bxy(4,1) = x4
bxy(4,2) = y4

flcoef(6,4) — Coefficients of the equations of the lines of intersection of the flow region boundary planes with the given fracture plane, zplane, in the local coordinate system of the particular fracture, calculated in subroutine RFLOFRC:
flcoef(6,3) = frcoef(6,3) * rotf(3,3)
(with z=zplane) and a'x + b'y + c'zplane + d = 0
for 1=1,6
flcoef(i,1) = a'
flcoef(i,2) = b'
flcoef(i,3) = c' * zplane + d
flcoef(i,4) = d

ifiro(maxcub) — key to array ninto, set up and used in subroutine DISCOUT, ic=1,ncub
ifiro(ic) = number of first entry in array ninto containing the number of an intersection

outside flow region which is on a fracture intersecting cube ic

ifirsi(maxfrc) — Reference number of first intersection found on

a fracture. Set up and used in subroutine INTERS to keep track of intersections found before a fracture is considered n=1,nfrac
ifirsi(n) = reference number of first intersection

ifrcu(2*maxfrc) Fracture/cube catalog set up and used in subroutine SUBINT, keyed by arrays jfirst, jlast and nextc,
i = 1, nentry
ifrcu(i) = number of the fractures in entry i

ilasi(maxfrc) Reference number of last intersection found on a fracture. Set up and updated in INTERS each time an intersection is found. Used to update local array nexti.

ilaso(maxcub) Key to array ninto, set up and used in subroutine DISCOUT, ic=1,ncub
ilaso(ic) = number of last entry in array ninto containing the number of an intersection outside flow region which is on a fracture intersecting cube ic.

jfirst(maxcub) Key to array ifrcu set up and used in subroutine SUBINT, ic=1,ncub
jfirst(ic) = number of first entry in array ifrcu containing the number of a fracture intersecting cube ic.

jlast(maxcub) Key to array ifrcu, set up, updated and used in subroutine SUBINT, ic=1,ncub
jlast(ic) = number of last entry in array ifrcu containing the number of a fracture intersecting cube ic

keybl(6,6) Key to intersections in bline array, entry (i,j) equals the row in bline containing (x,y) of intersection of sides i and j, or zero if lines are parallel or if plane i or j is parallel to fracture plane. Values originally set in subroutine BNDINT. After calling subroutine BNDSEG, keybl is effectively reduced to (6,2) to key the endpoints of the boundary segment on each side. Entries will be zero for any side not forming a boundary in the fracture plane.

bline(12,1)

$$\begin{bmatrix} x15 & y15 \\ x16 & y16 \\ x12 & y12 \\ x14 & y14 \\ x35 & y35 \\ x36 & y36 \\ x32 & y32 \\ x34 & y34 \\ x52 & y52 \\ x54 & y54 \\ x62 & y62 \\ x64 & y64 \end{bmatrix}$$

keybl(6,6)

$$\begin{bmatrix} 0 & 3 & 0 & 4 & 1 & 2 \\ 3 & 0 & 7 & 0 & 9 & 11 \\ 0 & 7 & 0 & 8 & 5 & 6 \\ 4 & 0 & 8 & 0 & 10 & 12 \\ 1 & 9 & 5 & 10 & 0 & 0 \\ 2 & 11 & 6 & 12 & 0 & 0 \end{bmatrix}$$

nextc(2*maxfrc) — Key to array ifrcu, set up and used in subroutine SUBINT,
i=1, nentry
nextc(i) = number of the next entry in array
ifrcu containing the number of a fracture
intersecting the same cube as fracture in entry i.

nexti(maxint) — Reference number of next intersection on fracture,
set up and used in subroutine INTERS to keep track
of intersections found before a fracture is
considered i=1,nint
nexti(i): number of next intersection on same fracture
as intersection number i

nexto(maxout*3) — Key to array ninto, set up and used in subroutine DISCOUT,
i=1, nentry
nexto(i) = number of the next entry in array
ninto containing the number of an intersection outside
flow region linked to the same cube as the one in entry i

ninto(maxout*3) — Intersection outside flow region/cube catalog set up and used in
subroutine DISCOUT, keyed by arrays ifiro, ilaso and nexto,
i=1,nentry
ninto(i) = number of the intersection outside
flow region in entry i.

ntrunc(7) — Truncation codes for the six boundary lines as well as the
total number of lines intersected by fracture nf,
local to subroutines TRUNCF, FLAG, FLAGNT and
BNDINT, BNDNOD, and AREAT; for i=1, 6
ntrunc(i) =
-1 fracture plane parallel to boundary plane
0 no intersection on side i
1 intersection at one point
2 intersect., frac. center inside boundary
3 intersect., fract. center outside boundary
4 intersect. through fracture center
ntrunc(7) = number of boundary lines intersected by
fracture nf

numfrc(maxcub) — Number of fractures intersecting each cube.
Computed and printed in subroutine WDIMES
ic = 1,ncub
numfrc(ic) = number of fractures intersecting
cube ic

numinc(maxcub) Total number of intersections (boundary + internal) on all fractures intersecting each cube. Computed and printed in subroutine WDIMES, ic=1,ncub
numinc(ic) = number of intersections on fractures intersecting cube ic

nxy(6,2) Codes for endpoints in axy, used by subroutine AREAT, values set in subroutine BNDNOD, i=1,6
nxy(i,1) = code for (x1,y1)
nxy(i,2) = code for (x2,y2)
code = 1 if point is inside circle
code = 2 if point is on circle

rot(3,4) Flow region rotation matrix and axes intercepts, local to subroutine ROFLOW
rot(1,1)= lambda 1 = cos(phi)*cos(theta)
rot(1,2)= lambda 2 = -cos(phi)*sin(theta)
rot(1,3)= lambda 3 = -sin(phi)
rot(1,2)= mu 1 = sin(theta)
rot(2,2)= mu 2 = cos(theta)
rot(2,3)= mu 3 = 0
rot(3,1)= nu 1 = sin(phi)*cos(theta)
rot(3,2)= nu 2 = -sin(phi)*sin(theta)
rot(3,3)= nu 3 = cos(theta)
rot(1,4)= xmesh/2
rot(2,4)= ymesh/2
rot(3,4)= zmesh/2

rotf(3,3) Rotation matrix from the global XYZ system to a local system convenient to the fracture plane ax + by + cz + d = 0, calculated in subroutine ROFRAC
rotf(1,1)= (xc+ad)/dk
rotf(2,1)= (yc+bd)/dk
rotf(3,1)= (zc+cd)/dk
rotf(1,2)= m1
rotf(2,2)= m2
rotf(3,2)= m3
rotf(1,3)= a
rotf(2,3)= b
rotf(3,3)= c
where dk is the distance between points (xc,yx,zc) and (-ad, -bd, -cd), and m1, m2, and m3 are determined such that the matrix is orthogonal

Table A-3. FMG3D Subroutine Outline

Fracture system in generation region.

FMG3D	FRAGEN	RANCEN		
		NORDIS		
		LOGDIS		
		EXPDIS	GGUBFS*	
		NORDIS1		
		UNIDIS		
		FRASTA		
	EQPLAN			
	SUBINT	ROFRAC		
		RFLOFRC		
	WFRAC1			
	TRUNCG			
	WFRAC1			
	WRFIVE			
	PFS	STATOR		

Fracture system in flow region.

FMG3D	WRFIVE			
	ROFLOW			
	TRUNCF	ROFRAC		
		RFLOFRC		
		FLAG	FLAGNT	
		BNDINT	BNDPTS	SIMULL
			COMSEG	
			BNDSEG	COMSEG
		BNDNOD	ENDPTS	SIMULC
			COMSEG	
		AREAT	CIRCPT	
	DISCAR			
	WFRACI			
	WINTER			
	PFS	STATOR		
	STAT2D			

Fracture intersections and conducting fracture system.

FMG3D	INTERS	ROFRAC		
		RFLOFRC		
		ROJTOF		
		INTFRC	SIMULC	
			COMPT	
			COMSEG	
		TRUNCN	SIMULL	
			COMPT	
		RFRCGLO		
	WINTER			

	INTFRA	
	DISCOUT	
	DISCAR	
	WFRAC1	
	WINTER	
	PFS	STATOR
	WDIMES	

Input to fluid flow model.

FMG3D	NODELE	NODSEG
	WNODES	

*International Mathematical and Statistical Library (IMSL) Subroutine

Table A-4. Description of FMG3D Subroutines

Subroutine	Description
AREAT	Computes the area of the fracture that lies outside the flow region and subtracts it from the area of the fracture. given: dk,rf,flcoef(6,4),ntrunc(7),axy(6.4),nxy(6.2) returns: flarea and error flag ierr uses subroutine: CIRCPT
BNDINT	Determines the 12 points of intersection of the six boundary lines in the fracture plane and from these determines the six or fewer line segments that form the flow region in the fracture plane. given: flcoef(6,4),iside(3,2)ntrunc(7),nf returns: bline(12,2),keybl(6,6),iflag uses subroutines: BNDPTS SIMULL COMSEG BNDSEG COMSEG
BNDNOD	Determines fracture/boundary intersections and endpoints of boundary nodes. given: ifrac,ifold,flcoef,(6,4),dk,rf,zplane,ntrunc(7), bline(12,2),keybl(6,6),kib returns: binte(6,maxint),ibkey(nfrac,2), ibside(2,maxint),ibinte(2,maxint)nfracb and the same endpoints for subroutine areat, i.e. axy(6,4),nxy(6,2) uses subroutines: ENDPTS SIMULC COMSEG
BNDPTS	Sets up input parameters and calls SIMULL to determine the point of intersection, (x,y), of lines ln1 and ln2. given: flcoef(6,4),ln1,ln2,nf returns: x,y,npara uses subroutines: SIMULL
BNDSEG	Given the 12 points of intersection of the six boundary lines, determines the six or fewer points, i.e., line segments, that define the flow region. given iside(3,2),bline(12,2),keybl(6,6) returns: number of boundary points nbp, error flag ierr, and adjusted keybl using only the first two columns

uses subroutine: COMSEG

CIRCPT — Determines the point where a line, from the center of a given circle through a given point inside the circle intersects the circle.

given: bxy(j,1),bxy(j,2),dk,rf,j where j=1 or 2
returns: bxy(j+2,1),bxy(j+2,2), and error flag ierr

COMPT — Given three points on the same line, determines whether the third is between the other two, coincident with one of them or not between them.

given: xy(n,2),n,np1,np2,np3
returns: loc (code to the location of np3)

COMSEG — Given four points in the xy array which define two line segments on a given line, determines which points, if any, define a common segment. The np array keys the two or fewer endpoints of the common segment.

given: xy(n,2),n,iopt, and endpoint key np(4)
returns: adjusted np(4), number of endpoints nn (equal to 0, 1, or 2), and error flag ierr

DISCAR — Eliminates fractures flagged for discarding by permuting the reference arrays, and adjusts the number of fractures

given: nfrac, nflag(nfrac),iref(nfrac), inew(nfrac), isetfr(nfrac), iseti(nset,8)
returns: adjusted nfrac, iref, inew,iseti

DISCOUT. — Among the intersections outside the flow region, which are recorded only for plotting purposes, discards the ones which lay on fractures flagged for discarding, and sorts them by cubes.

given: nfrac, ncub, nout, ifout(nout,2), nflag(nfrac), endout(6,nout), icub(ilengc), icubk(nfrac)
returns: updated ifout, endout, nout, and ifro(3*maxout), ifrok(ncub+1)

ENDPTS — Sets up inputs and calls SIMULC to find the points of intersection of side is and fracture circle nf.

given: flcoef(6,4),is,dk,rf
returns: nroots (the number of roots) and rows 3 and 4 of xy(4,2)
uses subroutine: SIMULC

EQPLAN — Determines the coefficients of the fracture plane equations from orientation angles and fracture centers.

given: maxfrc,nfrac,frac(i,nfrac),i=1 to 8
returns: frac(i,nfrac),i=9 to 12

EXPDIS — Generates random variables distributed exponentially with expected value ev.

given: maxfrc,k,n1,n2,dseed,ev
returns: (frac(k,i),i=n1,n2)
uses subroutine: GGUBFS

FLAG — Compares the position of the fracture with the six flow region boundary planes and flags the fracture for discarding, saving as is or possible truncation.

given: flcoef(6,4),zplane,dk,rf iside(3,2),itole
returns: iflag,ntrunc(7)
uses subroutine: FLAGNT

FLAGNT — Determines the truncation code for a given side, ln, which is a perpendicular distance dn from the fracture center.
Note: idn=iround(nd,itole), irf=iround(rf,itole), etc.

given: ln,idn,irf,idm,idiff
returns: ntrunc(ln) and adjusted ntrunc(7)

FRAGEN — Reads or generates fracture centers and characteristics.

given: maxfrc,maxset
reads: iranf,dseed,title(40),rgene,nsets,itole,
iseti(nsets,8),rseti(nsets,16),
returns: nfrac and frac(i,nfrac) i=1 to 8 [read or generated]
uses subroutines: RANCEN EXPDIS FRASTA
NORDIS NORDIS1
LOGDIS UNIDIS

FRASTA — Calculates the mean and standard deviation of a set of random variables.

given: maxfrc,k,n1,n2,frac
returns: ev and sd of (frac(k,i),i=n1,n2)

GGUBFS — An IMSL, (International Mathematical and Statistical Library) function, generates pseudo-random numbers uniformly distributed between zero and one.

given: dseed
returns: ggubfs, and a new value of dseed

IROUND — Round-off function.

given: a,nplace

returns: iround = integer part of a*10**nplace

INTERS Rotates fracture ifrac, flow region boundary planes, and fracture jfrac to a system convenient to plane ifrac then determines fracture/fracture intersections and truncates them at flow region boundaries. Boundary nodes and fracture intersections are then rotated back to the global XYZ coordinate system. Truncated intersections are recorded for plotting.

given: maxfrc,maxint,nfrac,nfracb,frac(12,nfrac),iref(nfrac),
inew(nfrac),ibkey(nfrac,2),ninb,itole,frcoef(6,4),
iside(3,2),binte(6,maxint),ibside(2,maxint)
returns: intel(6,maxint),inte2(2,maxint),inte3(maxint*2+1),
inside(2,maxint),inkey(nfrac,2),nint,endout(6,maxout),
ifout(maxout,2) and adjusted iref and inew
uses subroutines:

ROFRAC	
RFLOFRC	
ROJTOF	
INTFRC	SIMULC
	COMPT
	COMSEG
TRUNCN	SIMULL
	COMPT
RFRCGLO	

INTFRA Determines non-conducting fractures, (i.e. those with fewer than two intersections, internal and/or boundary), eliminates them from intersection arrays, and flags them for discarding.

given: maxfrc,maxint,nfrac,ibkey(nfrac,2),ninb,
intel(6,maxint),inte2(2,maxint),iref(nfrac),inew(nfrac),
inte3(maxint*2+1),inkey(nfrac,2),inside(2,maxint),nint
returns: adjusted intel,inte2,inte3,inkey,inside, and
ibkey arrays, adjusted nint and ninb,
nflag(nfrac),ndiscd,inst

INTRFC Determines intersection, if any, between fractures nf and nj given fracture circle (x-dk)**2 + y**2 = rf**2 in zplane, sphere nj with center (x,y,z) and radius r, and the line of intersection of planes nf and nj, ax + by + c = 0.

given: nf,dk,zplane,rf,nj,a,b,c,x,y,z,r,itole
returns: intersection endpoints in rows 3 and 4 of xy(4,2) and
ntf, (ntf=1 -- intersection, ntf=2 -- no intersection)
uses subroutines: SIMULC
COMPT
COMSEG

LOGDIS Generates random variables distributed lognormally with expected value

ev and standard deviation sd.

given: maxfrc,k,n1,n2,dseed,ev,sd
returns: (frac(k,i),i=n1,n2)
uses subroutine: GGUBFS

NODELE Creates nodal arrays from boundary node and fracture intersection arrays. Calls NODSEG to discretize nodes.

given: maxfrc,maxint,nfrac,inte1,(6,maxint),iref(nfrac),inew(nfrac),
inte2(2,maxint),inte3(maxint*2+1),inside(2,maxint),
ibkey(nfrac,2),ninb,inkey(nfrac,2),nint,inst,nseg,nsegb
returns: node1(maxint,6),node2(maxint,2),nside(maxint,2),
node3(maxint*2+1),nkey(nfrac,2),nnode
uses subroutine: NODSEG

NODSEG Divides the line segment determined by (inte1 (i,n),i=1,6) into nseg (internal nodes), or nsegb (boundary nodes), equal segments. The endpoints are stored in rows n1 through n2 of array node1 and the corresponding side codes are stored in array nside. If iopt = 1, the line segment is a boundary node, if iopt = 2, it is a fracture intersection.

given: maxint,inte1(6,maxint),inside(2,maxint),n,nseg
or nsebg,n1,iopt
returns: ((node1(i,j),j=1,6),i=n1,n2),
((nside(i,j),j=1,2),i=n1,n2), n2

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

given: maxfrc,k,n1,n2,dseed,ev,sd
returns: (frac(k,i),i=n1,n2)
uses subroutine: GGUBFS

NORDIS1 Generates random variables distributed normally with expected value ev and standard deviation sd where ev is proportional to the logarithm of another parameter, (i.e., aperture is proportional to log. of radius).

given: maxfrc,k,n1,n2,dseed,ycept,slope,sd
returns: (frac(k,i),i=n1,n2)
uses subroutine: GGUBFS

PFS Computes and prints statistics on characteristics of each fracture set.

given: maxfrc,maxset,nsets,iseti(nsets,8),iref(nfrac),ncall,
isetfr(nfrac),rseti(nsets,16),frac(12,nfrac),iray,idate,
rgene,xmesh,ymesh,zmesh,rophi,rothe
returns: writes fracture statistics on unit 2 for printout
uses subroutine: STATOR

RANCEN Generates random fracture centers within generation sphere.

given: maxfrc,n1,n2,dseed,rgene,itole
returns: ((frac(j,i),j=5,7),i=n1,n2)
uses subroutines: GGUBFS

RFLOFRC Rotates flow region boundary planes to local fracture system and returns coefficients of the lines of intersection with the fracture plane.

given: frcoef(6,4),rotf(3,3),zplane
returns: flcoef(6,4)

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

given: isiz1,isiz2,xyz(isize1,isiz2),rotf(3,3),k1,k2,inv
returns: transformed (xyz(k,isiz2),k=k1,k2)

ROFLOW Computes flow region rotation matrix, axes intersections, and coefficients of the equation of the flow region boundary planes.

given: xmesh,ymesh,zmesh,rophi,rothe,iside(3,2)
returns: frcoef(6,4)

ROFRAC Computes the matrix of rotation from the global system to a local system determined by the fracture being investigated, computes the new z-plane and fracture center (dk,0,zplane).

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

ROJTOF Rotates coefficients of plane jfold and coordinates of the center of fracture jfold from the global coordinate system to the local system of fracture ifold defined by rotation matrix rotf and zplane; determines line of intersection between planes ifold and jfold.

given: maxfrc,jfold,frac(12,nfrac),rotf(3,3),zplane
returns: a,b,c,d,x,y,z

SIMULC Solves simultaneous equations of line ax + by + c = 0 and circle (x-xc)**2 + (y-yc)**2 = r**2, and determines the number of real roots, nroots.

given: a,b,c,xc,yc,r

returns: nroots,x1,y1,x2,y2

SIMULL — Solves the simultaneous equations of lines with coefficients a1,b1,c1, and a2,b2,c2. If lines are parallel, npara=2.

given: a1,b1,c1,a2,b2,c2
returns: x,y, and npara

STATOR — Calculates mean and angular deviation of a set of orientations

given: nsets,iseti(nsets,8),rseti(nsets,16)
isetfr(nfrac),iref(nfrac),frac(12,nfrac)
returns: eval(nsets,i),sdev(nsets,i),i=1 to 2

STAT2D — Calculates and prints length statistics on the traces of the fractures on the sides of the flow region.

given: nfrac,nfracb,nflow,iref(nfrac),isetfr(nfrac),
intel(6,maxint),inte2(2,maxint),ibkey(nfrac,2)
calculates and prints fracture length statistics on unit 2

SUBINT — Divides the generation region into cubic subregions, and assigns fractures to the cubes they intersect. The initialization of iref(nfrac) and inew(nfrac) is also done in SUBINT.

given: nfrac,iside,itole,iord,rgene,frac(12,nfrac)
returns: ncub,ilengc,icub(ilengc),ifcu(ilengc),
icubk(nfrac),ifcuk(ncub)
uses subroutines: ROFRAC
RFLOFRC

TRUNCF — For fracture ifrac, where ifrac=1,nfrac, determines flow region boundary nodes (fracture/boundary intersections) and truncates the fracture if partially outside the flow region, or codes the fracture for discarding or saving as is. Reference arrays are adjusted to put boundary fractures first.

given: maxfrc,maxint,frac(12,nfrac),frcoef(6.4),iref(nfrac),
inew(nfrac),nfrac,iside(3,2),itole
returns: nflag(nfrac),ibkey(nfrac,2),binte(6,maxint)
ibside(2,maxint),ibinte(2,maxint),ninb,adjusted
iref and inew and truncated areas frac(8,nfrac)
uses subroutines:

ROFRAC		
RFLOFRC		
FLAG	FLAGNT	
BNDINT	BNDPTS	SIMULL
	COMSG	
	BNDSEG	COMSEG

BNDNOD	ENDPTS	SIMULC
	COMSEG	
AREAT	CIRCPT	

TRUNCG Determines which fractures extend beyond the generation region boundary and truncates them at that boundary.

given: maxfrc,nfrac,rgene,frac(12,nfrac),iref(nfrac),itole
returns: frac(8,nfrac),flag(nfrac)

TRUNCN Truncates the intersection of fractures ifold and njold at any or all of the nbn boundary nodes on fracture nfold, and keeps track of the truncated segments.

given: maxfrc,maxint,nfold,flcoef(6,4),iside(3,2),nbn,
binte(6,maxint),ibside(2,maxint),ibkey(maxfrc,2),
aj,bj,cj,xy(4,2)
returns: truncated xy(4,2),nside(2),endout(6,nout),
ifout(2,nout),nn,nout
uses subroutines: SIMULL
COMPT

UNIDIS Generates random variables distributed uniformly over (center-range, center+range).

given: maxfrc,k,n1,n2,dseed,center,range
returns: (frac(k,i),i=n1,n2)
uses subroutine: GGUBFS

WDIMES Writes the fractures and intersections characteristics on unit 8 for input to the plotting program DIMES.

given: xgene,ygene,zgene,xmesh,ymesh,zmesh,rophi,rothe,inst,
nint,nfrac,nout,iord,ifcuk(ncub),ifcu(ilengc),
iref(nfrac),inew(nfrac),ibkey(nfrac,2),inkey(nfrac,2),
frac(12,nfrac),ifrok(ncub+1),endout(6,nout),ifout(2,nouk),
intel(6,maxint),inte2(2,maxint),inkey(2,maxint)
writes all mesh characteristics on unit 8.

WFRAC1 Writes fracture characteristics of each fracture set on unit 4 for printout. Output is controlled by iopt:

iopt=1 - all generated fractures
iopt=2 - truncated fractures in generation region
iopt=3 - truncated fractures in flow region
iopt=4 - fractures in flow region after elimination
of nonconducting fractures

given: iopt,maxfrc,nfrac,frac(12,nfrac),rgene,iray,idate,

maxset,nsets,iseti(nsets,8),iref(nfrac),inew(nfrac),
nflag(nfrac),rseti(nsets,16),xmesh,ymesh,zmesh,
rophi,rothe
returns: output file on unit 4

WINTER Writes boundary nodes in local fracture coordinates (all fractures in flow region), if iopt=1, or writes boundary nodes and fracture intersections in global coordinates (after elimination of nonconducting fractures), if iopt=2, on unit 4 for printout.

given: maxfrc,maxint,nfrac,intel,(6,maxint),inte2(2,maxint),
inside(2,maxint),ibkey(nfrac,2),iref(nfrac),ninb,inkey(nfrac,2),
nint,inst,inte3(maxint*2+1),inew(nfrac),iopt
returns: output file on unit 4

WNODES Writes flow region, fracture and nodal parameters on unit 6 for input to fluid flow model, DISCEL.

given: maxfrc,maxint,frcoef(6,4),nfrac,frac(12,nfrac),nnode,
iref(nfrac),inew(nfrac),node1(6,maxint),nside(2,maxint)
node2(2,maxint),node3(maxint*2+1),nkey(nfrac,2),iray,
idate,title(40),xmesh,ymesh,zmesh,rophi,rothe
returns: output file on unit 6

WRFIVE iopt=1 - writes all primary fracture system data on unit 5,
iopt=2 - reads all primary fracture system data from unit 5

given: maxfrc,maxset, iopt
data written or read: iray,idate,title(40),itole,nsets,
iseti(nsets,8),rseti(nsets,16),nfrac,rgene,
frac(12,nfrac),isetfr(nfrac),nflag(nfrac),icubk(nfrac),
ncub,ilengc,ifcuk(ncub),icub(ilengc),ifcu(ilengc)

APPENDIX B

Listing of Program FMG3D

include files for FMG3D:

```
file COMBINT.BLKF
     real*4 binte(10,maxint+1)
     integer*4 ibside(10,maxint),ibinte(10,maxint)
     common/inter/binte,inte3(maxint*2+1),
    2             ibkey(maxfrc,2),inkey(maxfrc,2),
    3             ninb,nint,inst,nfracb
     equivalence (binte(10,1),ibinte(4,1),ibside(2,1))

file COMFINT.BLKF
     real*4 inte1(10,maxint+1)
     integer*4 inte2(10,maxint),inside(10,maxint)
     common/inter/inte1,
    1             inte3(maxint*2+1),ibkey(maxfrc,2),
    2             inkey(maxfrc,2),ninb,nint,inst,nfracb
     equivalence (inte1(10,1),inte2(4,1),inside(2,1))

file COMFLO.BLKF
     common/floreg/xmesh,ymesh,zmesh,rophi,rothe

file COMFRACN.BLKF
     common/fract/frac(12,maxfrc),new(maxfrc)

file COMFRC.BLKF
     common/fract/frac(12,maxfrc),nflag(maxfrc),
    '             isetfr(maxfrc)

file COMNOD.BLKF
     real node1(maxnod,6)
     common/nodes/node1,node2(maxnod,2),nside(maxnod,2),
    1             node3(maxnod*2+1),nkey(maxfrc,2),nnode

file COMOUT.BLKF
     common/intout/endout(6,maxout),rgene,ifout(maxout,2),
    '              ifrok(maxcub+1),ifro(3*maxout),nout

file COMREF.BLKF
     common /refer/iref(maxfrc),inew(-6:maxfrc)

file COMSET.BLKF
     common/sets/rseti(maxset,16),iseti(maxset,8),nsets

file COMSUBI.BLKF
     integer*2 icub(maxf2)
     common /subint/ icub,ilengc,
    '                ifcu(maxf2),ifcuk(maxcub),
    '                icubk(maxfrc),iord,ncub

file MAXDIM.BLKF
     parameter (maxfrc=50000,maxset=5,maxint=50000,
    1           maxnod=50000,maxout=10000,maxord=6)
     parameter (maxf2=maxfrc*2,maxcub=maxord**3)
```

```
c
      PROGRAM FMG3D
c
c ***************************************************************
c
c ******    Three-Dimensional Fracture Generation Model    ******
c
c *** Generates Fracture Network for Fluid Flow Model DISCEL ***
c
c ********                  Written By                    ********
c
c ************       PEGGY (MACLEAN) GILMOUR         ************
c
c ****************             for              ****************
c
c ************               JANE LONG              ************
c
c
c ***************************************************************
c
c
c ************************* S E T U P ***************************
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      include 'comset.blkf/list'
      include 'comnod.blkf/list'
      include 'comflo.blkf/list'
      include 'comfint.blkf/list'
      include 'comout.blkf/list'
      include 'comref.blkf/list'
      include 'comsubi.blkf/list'
      dimension frcoef(6,4),iside(3,2)
c     equivalence (binte,inte1),(ibside,inside),(ibinte,inte2)
      integer title(40)
      character idate*9,iray*7
      real*8 dseed
      data iside/1,5,2,3,6,4/
c
c
c *** Request job information.
        iray = 'fmg3d  '
        call DATE (idate)
      open(unit=1,file='fmg3d.inp',status='old',readonly)
      open(unit=2,file='fmgpr.dat',status='new',shared)
      open(unit=6,file='discel.dat',status='new',shared)
c
c *** Print title page
      write(2,510)
c
c
c ****** I N P U T   C O N T R O L   V A R I A B L E S *******
c
      read(1,490)icont,iplot,imesh
      read(1,490)iplane,idump,iord
      write(2,520)icont,iplot,imesh,
     '            iplane,idump,iord
c
c     if(iplot.eq.1)
c    '  open(unit=3,file='fmg3d.plt',status='new',shared)
```

```
      if(iplane.eq.1)
     '  open(unit=7,file='plane.inp',status='new',shared)
      if(idump.eq.1)
     '  open(unit=4,file='fmg3d.dmp',status='new',shared)
      if(iord.eq.0)iord=1
      if (icont.eq.3) go to 185
c
c
c ********** F R A C T U R E   G E N E R A T I O N **********
c
c
c *** Call subroutine FRAGEN to read in or generate fracture
c *** characteristics and store them in array frac
c
  110 call FRAGEN (nfrac,itole,iranf,dseed,title,rgene)
c
c
c *** Call subroutine EQPLAN to compute the coefficients of the
c *** fracture plane equations.
c
  120 call EQPLAN(nfrac)
c
c     call subroutine SUBINT to define subregions for
c     use in OVERLAP and INTERS.
c
      call subint(nfrac,iside,itole)
c
c
c *** Call subroutine OVRLAP to find and combine overlapping
c *** fractures lying in the same plane.
c
c     call OVRLAP(nfrac,itole,dseed)
c
c
c *** Call WFRAC1 to print fracture characteristics by sets, iopt=1.
c
      if(idump.eq.1)call WFRAC1(1,nfrac,rgene,iray,idate)
c
c
c
c *** Call subroutine TRUNCG to truncate any fractures extending
c *** beyond the boundary of the generation sphere.
c
      call TRUNCG (nfrac,rgene,itole)
c
c
c *** Call WFRAC1 to print fracture characteristics by sets, iopt=2.
c
      if(idump.eq.1)call WFRAC1 (2,nfrac,rgene,iray,idate)
c
c
c *** Call WRFIVE to write primary fracture system data to unit 5.
c
      call WRFIVE (1,iray,idate,title,itole,nfrac,rgene)
c
c
c *** Call PFS to calculate and print fracture statistics
c
      call PFS (nfrac,nflow,iray,idate,rgene,1)
c
```

```
c
  180 if (icont.ne.1) go to 185
      write(2,370)
      stop
c
c
c
c ******************** F L O W   R E G I O N ********************
c
c
c *** Read flow region and nodal segmentation parameters.
c
  185 nflow=0
c
  190 read (1,660,end=191) xmesh,ymesh,zmesh,rophi,rothe
      read (1,495,end=191) nseg,nsegb
c
c     increment number of flow regions
c
      nflow=nflow+1
c
      go to 193
c
  191 write(2,380)
c *** end of input data
      stop
c
c
c *** Call WRFIVE to read all primary fracture system data from
c *** unit 5.
c
  193 call WRFIVE (2,iray,idate,title,itole,nfrac,rgene)
c
      if ((xmesh**2+ymesh**2+zmesh**2)/4.0 .le. rgene**2) go to 195
      write(2,385) xmesh,ymesh,zmesh,rgene
      go to 190
c
c
c *** Call subroutine ROFLOW to compute the coefficients of
c *** the equations of the flow region boundary planes.
c
  195 call ROFLOW(iside,frcoef)
c
c
c *** Write plot file identification, if iplot =1.
c
c     if (iplot .eq. 0) go to 197
c     write(3,800) iray,idate
c     write(3,810) (title(i),i=1,40)
c     write(3,820) xmesh,ymesh,zmesh,rophi,rothe,nfrac
c
c
c *** Call subroutine TRUNCF to truncate each fracture that intersects
c *** the flow region boundary or set the discard code for any fracture
c *** that lies outside the flow region.
c
  197 call TRUNCF(frcoef,nfrac,iside,itole)
c
c
c *** Call subroutine DISCAR to eliminate fractures flagged as lying
```

```
c *** entirely outside the flow region.
c
      nout=0
      call DISCAR (nfrac)
c
      if (nfrac.gt.0) go to 200
c *** There are no fractures in this flow region, go to next set of
c *** flow region parameters.
      write(2,390) rophi,rothe
      go to 190
c
c
c *** Call WFRAC1 to print fracture characteristics by sets, iopt=3.
c
  200 if(idump.eq.1)call WFRAC1 (3,nfrac,rgene,iray,idate)
c
c
c
c *** Call WINTER to print boundary nodes in local
c *** fracture coordinates
c
      if(idump.eq.1)call WINTER (nfrac,1)
c
c
c
c *** Call PFS to calculate and print fracture statistics
c
      call PFS (nfrac,nflow,iray,idate,rgene,2)
c
c     call STAT2D to calculate and print trace length
c     statistics on sides of flow region
c
      call STAT2D(nfrac,nflow)
c
c
c ****************** FRACTURE INTERSECTIONS *****************
c
c
c *** Call subroutine INTERS to determine fracture/fracture
c *** intersections and truncate them at flow region boundaries.
c
  250 call INTERS (nfrac,itole,frcoef,iside)
c
c
c *** If iplane = 1, write fracture network information on
c *** unit 7 for Program IPLANE.
c
      if(iplane.eq.1)then
        write(7,700) iray,idate
        write(7,710) (title(i),i=1,40)
        write(7,720) xmesh,ymesh,zmesh
        write(7,730) rophi,rothe
  700   format (a7,5x,a9)
  710   format (20a4)
  720   format ('    xmesh ',f10.4,'    ymesh ',f10.4,
     1          '    zmesh ',f10.4)
  730   format ('    rophi ',f10.4,'    rothe ',f10.4)
        inst=ninb
        call WINTER(nfrac,4)
       endif
```

```
      if(idump.eq.1)call WINTER(nfrac,3)
c
c
c *** If iplot=1, call WPLOT to write a flow region plot file on unit 3.
c *** (This file must be written after boundary nodes have been
c *** rotated from local to global coordinates in sub. inters and
c *** before discarding non-conducting fractures.)
c
c 255 if (iplot .eq. 1) call WPLOT(1,nfrac)
c
c
c *** Call subroutine INTFRA to find non-conducting fractures,
c *** eliminate them from intersection arrays, and flag the fractures
c *** for discarding.
c
      call INTFRA (nfrac,ndiscd)
c
c
c *** call subroutine DISCOUT to discard intersections outside
c *** the flow region and laying on a discarded fracture.
c
      call DISCOUT(nfrac)
c
c *** Call subroutine DISCAR to eliminate non-conducting fractures
c *** flagged by intfra.
c
      if (ndiscd .eq. 0) go to 260
      call DISCAR(nfrac)
c
      if (nfrac .gt. 0) go to 260
c *** There are no conducting fractures in this flow region, go to
c *** next set of flow region parameters.
      write(2,395) rophi,rothe
      go to 190
c
c
c *** Call WFRAC1 to print fracture characteristics by sets, iopt=4.
c
  260 if(idump.eq.1)call WFRAC1 (4,nfrac,rgene,iray,idate)
c
c
c *** Call WINTER to print boundary nodes and fracture/fracture
c *** intersections, iopt=2
c
      if(idump.eq.1)call WINTER (nfrac,2)
c
c
c *** Call PFS to calculate and print fracture statistics
c
      call PFS (nfrac,nflow,iray,idate,rgene,3)
c
c
c *** if iplot = 1, call WDIMES to write an input file for DIMES
c
      if(iplot.eq.1) call WDIMES (nfrac,title,nflow)
c
c
  320 if (imesh .ne. 0) go to 325
      write(2,410)
      stop
```

```
c
c
c
c ************************ N O D E S ***************************
c
c
c *** Check nodal segmentation parameters.  If input
c *** incorrectly, set to default values.
c
  325 if (nseg .gt. 0) go to 327
      write(2,400) nseg
      nseg=1
      nsegb=3
      go to 330
  327 if (nsegb .lt. nseg) nsegb=nseg+2
  330 write(2,530) nseg,nsegb
c
c
c *** Call subroutine NODELE to fill nodal arrays for DISCEL
c
      call NODELE (nfrac,nseg,nsegb)
c
c
c *** Call subroutine SORT to renumber nodes for reduced bandwidth
c
c     call SORT (**********************
c
c
c *** Call WNODES to print nodal arrays and write input file for DISCEL
c
      call WNODES (frcoef,nfrac,nseg,nsegb,iray,idate,title)
c
c
c *** if iplot = 1, write an input file for a plotting program on unit 3
c
c     if (iplot .eq. 1) call WPLOT(2,nfrac)
c
c

c *** Go to 190 to read next flow region
c *** or end of input data.
c
      go to 190
c
c
c
c ************ F O R M A T   S T A T E M E N T s *************
c
c
c
  370 format(1h0,27hnormal program stop,icont=1)
  380 format(1h0,32hnormal program stop,end of input)
  385 format(1h1,'Flow region exceeds limit of generation region.',/,
     1 1x,6hxmesh=,f8.4,8h  ymesh=,f8.4,8h  zmesh=,f8.4,8h  rgene=,
     2 f8.4,/,1x,30h*** go to next flow region ***,////)
  390 format(1h0,48hThere are no fractures in flow region for rophi=,
     1        f6.2,8h, rothe=,f6.2)
  395 format(1h0,44hthere are no conducting fractures for rophi=,f6.2,
     1         8h, rothe=,f6.2)
```

```
  400 format(1h0,12hError, nseg=,i4,//,1x,16hnseg set to one,,
     1 19h nsegb set to three,//)
  410 format(1h0,33hnormal end of generation,imesh =0)
  490 format(10x,i5,2(15x,i5))
  495 format(10x,i5,15x,i5)
  510 format(1h1,5(/),10x,10(6h******),///,14x,12hT H R E E   ,
     1 40hD I M E N S I O N A L   F R A C T U R E ,///,24x,
     2 32hG E N E R A T I O N   M O D E L ,///,35x,10hF M G 3 D ,
     3 ///,30x,20hw r i t t e n   b y ,///,
     4 17x,46hP E G G Y   ( M A C L E A N )   G I L M O U R ,
     5 ///,10x,10(6h******),20(/))
  520 format(1h0,15hprogram options,//,
     1 1x,7hicont =,i5,10x,7hiplot =,i5,10x,7himesh =,i5/
     2 1x,7hiplane=,i5,10x,7hidump =,i5,10x,7hiord  =,i5)
  530 format (1h1,34hEach fracture intersection will be,
     1 14h divided into ,i3,10h segments.,/,
     2 41h Each boundary node will be divided into ,i3,
     3 10h segments.)
  660 format (3(10x,f10.4))
c
  800 format (a7,5h --- ,a9)
  810 format (20a4)
  820 format (5f10.4,i10)
      end

c
c
c *** Round off function
c
      function iround (a,nplace)
      aplace=a*10.**nplace
      iround=nint(aplace)
      return
      end
      SUBROUTINE AREAT(dk,rf,flcoef,ntrunc,axy,nxy,flarea,ierr)
c
c *** This subroutine computes the portion of the fracture that
c *** lies within the flow region boundary.
c
      dimension flcoef(6,4),ntrunc(7),axy(6,4),nxy(6,2)
      dimension bxy(4,2)
      parameter (pi=3.141592654,dmin=0.005)
c
      at=0.0
      rfsq=rf*rf
      flarea=pi*rfsq
      nt3=1
      nt4=1
      j4=0
c
      do 99 i=1,6
      nt=ntrunc(i)
      if (nt.le.1) go to 99
      if (nt.eq.4) go to 40
c
c *** Store points p1,p2,p3,p4 in bxy(4,2) if ntrunc(i)=2 or 3.
c
      do 20 j=1,2
      bxy(j,1)=axy(i,2*j-1)
      bxy(j,2)=axy(i,2*j)
```

```
      if (nxy(i,j).eq.1) go to 10
c
c *** Point j is on the circle, point j+2 is the same as point j.
      bxy(j+2,1)=bxy(j,1)
      bxy(j+2,2)=bxy(j,2)
      go to 20
c
c *** Point j is not on the circle, determine point j+2.
   10 jc=j
      call CIRCPT(bxy,jc,dk,rf,ierr)
      if (ierr.eq.0) go to 20
      write(2,110) i,nt
      go to 100
c
   20 continue
      go to 80
c
c *** Boundary line i goes through the fracture center, ntrunc(i)=4.
c
   40 if (i.eq.j4) go to 99
      j4=i
   42 dist1=sqrt((axy(j4,1)-dk)**2+axy(j4,2)**2)
      dist2=sqrt((axy(j4,3)-dk)**2+axy(j4,4)**2)
      dist12=sqrt((axy(j4,1)-axy(j4,3))**2+(axy(j4,2)-axy(j4,4))**2)
      dist=dist1+dist2
      dd=abs(dist - dist12)
      if (dd .lt. dmin) go to 45
c
c *** Both points are on the same side of center.
      atos=0.0
      go to 90
   45 if (dist1 .lt. dmin) go to 50
      if (dist2 .lt. dmin) go to 55
c
c *** Points are on opposite sides of center.
      atos=0.5*pi*rfsq
      go to 90
c
c *** One point is on the center, (dk,0), store the other point.
   50 bxy(nt4,1)=axy(j4,3)
      bxy(nt4,2)=axy(j4,4)
      go to (65,60),nxy(j4,2)
c
   55 bxy(nt4,1)=axy(j4,1)
      bxy(nt4,2)=axy(j4,2)
      go to (65,60),nxy(j4,1)
c
c *** This point is on the circle.
   60 bxy(nt4+2,1)=bxy(nt4,1)
      bxy(nt4+2,2)=bxy(nt4,2)
      go to 70
c
c *** The point is not on the circle, determine the corresponding point.
   65 call CIRCPT(bxy,nt4,dk,rf,ierr)
      if (ierr.eq.0) go to 70
      write(2,110) i,nt
      go to 100
c
   70 if (nt4.eq.2) go to 80
      do 75 i1=i+1,6
```

```
      if (ntrunc(i1).ne.4) go to 75
      j4=i1
      nt4=2
      go to 42
   75 continue

c
c *** Compute the area of sector p3-center-p4 using the perpendicular
c *** distance from line p3p4 to center (dk,0).
c
   80 a=bxy(4,2)-bxy(3,2)
      b=bxy(3,1)-bxy(4,1)
      sqab=sqrt(a*a+b*b)
      if(sqab/rf.le.0.01)then
        theta=0.
       else
        c=bxy(4,1)*bxy(3,2)-bxy(3,1)*bxy(4,2)
        dist=abs(a*dk+c)/sqab
        if(dist.gt.rf)then
c         write(2,120)dist,i
c         ierr=1
c         return
          theta=0.
         else
          theta=acos(dist/rf)
         endif
       endif
      atos=rfsq*theta
      if (nt.lt.4) go to 85
c
c *** Two boundary lines intersect at (dk,0).
      atos=pi*rfsq-atos
      go to 90
c
c *** Compute the area of triangle p1-center-p2 and
c *** subtract from sector p3-center-p4.
   85 atri=0.5*abs(bxy(2,2)*(bxy(1,1)-dk)+bxy(1,2)*(dk-bxy(2,1)))
      atos=atos-atri
c
c *** Determine area to be truncated.
      if (nt.eq.3) go to 95
c
c *** Center on or inside boundary.
   90 at=at+atos
      go to 99
c
c *** Center outside boundary.
   95 if (nt3.eq.2) go to 97
      at=at+(pi*rfsq-atos)
      nt3=2
      go to 99
   97 at=at-atos
c
   99 continue
c
      flarea=flarea-at
  100 return
c
  110 format(1h0,//,' Error in sub. areat, side',i2,', ntrunc(i)=',
     1       i2,'.',/,' Less than two roots from simulc.')
```

```
  120 format(1h0,//,' Error in sub. areat, dist=',f10.3,', side=',
     1       i2)
c
      end
      SUBROUTINE BNDINT(flcoef,iside,iflag,ntrunc,nf,bline,keybl)
c
c *** Determine the 12 points of intersection of the 6 boundary
c *** lines in the plane of fracture nf, then determine the six
c *** or fewer line segments that form the flow region in the
c *** fracture plane.
c
      dimension flcoef(6,4),iside(3,2),ntrunc(7)
      dimension bline(12,2),keybl(6,6)
      dimension nl(4),npt4(4)
c
      nbp=0
      npt=0
      ipara=1
c
      do 20 k1=1,3
      do 20 k2=1,2
      ln1=iside(k1,k2)
c
      do 20 j1=k1,3
      do 15 j2=1,2
      ln2=iside(j1,j2)
      if (j1.eq.k1) go to 12
      npt=npt+1
      if (npt.le.12) go to 5
      write(2,110) nf,((flcoef(i,j),j=1,4),i=1,6)
      stop
c
    5 if (ntrunc(ln1).eq.-1 .or. ntrunc(ln2).eq.-1) go to 10
c
c *** Call BNDPTS to determine the point of intersection of l1 and l2.
c
      call BNDPTS(flcoef,ln1,ln2,nf,x,y,npara)
      go to (8,6),npara
c
    6 if (ipara.eq.2) go to 10
      kpara=k1
      jpara=j1
      ipara=2
      go to 10
c
    8 nbp=nbp+1
      bline(npt,1)=x
      bline(npt,2)=y
      keybl(ln1,ln2)=npt
      keybl(ln2,ln1)=npt
      go to 15
c
c *** Fracture plane is parallel to one pair of boundary planes
c *** or two pairs of boundary lines are parallel.
   10 bline(npt,1)=0.
      bline(npt,2)=0.
      keybl(ln2,ln1)=0
   12 keybl(ln1,ln2)=0
c
   15 continue
```

```
c
   20 continue
c
c
      go to (29,21),ipara
c
c *** Four boundary lines are parallel.  Determine which two,
c *** if any, are flow region boundaries in the fracture plane.
c
   21 lpara=iside(6-kpara-jpara,1)
      mpara=iside(6-kpara-jpara,2)
      do 22 i=1,2
      nl(i)  =iside(kpara,i)
   22 nl(i+2)=iside(jpara,i)
      do 23 i=1,4
   23 npt4(i)=keybl(lpara,nl(i))
      call COMSEG(bline,12,npt4,nn,1,ierr)
      if (ierr.eq.0) go to 24
      write(2,130) (nl(i),i=1,4),lpara,nf
      stop
c
   24 if (nn.eq.2) go to 25
      iflag=-1
      go to 100
c
   25 do 26 i=1,4
      if (npt4(i).ne.0) go to 26
      keybl(lpara,nl(i))=0
      keybl(mpara,nl(i))=0
      keybl(nl(i),lpara)=0
      keybl(nl(i),mpara)=0
   26 continue
      nbp=4
c
c *** Call BNDSEG to determine the endpoints of the line segments
c *** that form the flow region in the fracture plane.
c
   29 call BNDSEG(iside,bline,keybl,nbp,ierr)
      if (ierr.eq.0) go to 30
      write(2,120) nf
      stop
c
c *** If there is no flow region in this plane, discard the fracture.
   30 if (nbp.eq.0) iflag=-1
c
c
c
  100 return
c
  110 format (1h0,' error in bndint, fracture number ', i5,/,
     1        ' more than 12 points of intersection',/,
     2        ' flcoef=',/,6(4f15.4,/))
  120 format (1h0,' called by bndint, fracture',i4)
  130 format (1h0,' error in comseg called by bndint',/,1x,
     1 ' four parallel lines:',4(i3,','),' intersected by line',
     2  i3,/,1x,' fracture',i5)
c
      end
      SUBROUTINE BNDNOD(ifrac,ifold,flcoef,dk,rf,zplane,
     1   ntrunc,bline,keybl,kib,axy,nxy,iside)
```

```
c
c *** This subroutine determines which fracture/boundary
c *** intersections are to be boundary nodes in the finite
c *** element mesh.
c
      include 'maxdim.blkf/list'
      include 'combint.blkf/list'
      include 'comfrc.blkf/list'
      include 'comref.blkf/list'
      dimension flcoef(6,4),iside(3,2),ntrunc(7)
      dimension bline(12,2),keybl(6,6)
      dimension np(4),xy(4,2),axy(6,4),nxy(6,2)
c
c *** The line segment of intersection that lies within
c *** the flow region is determined and the endpoints
c *** stored in an array of boundary nodes, binte with
c *** ibkey(ifrac,2) as the key to the entries.  The endpoints
c *** are also stored in axy(6,4) for subroutine AREAT.
c
      nbn=0
      kbn=kib-1
c
      do 80 kpair=1,3
      do 80 kside=1,2
      is=iside(kpair,kside)
c
      do 5 j=1,2
    5 nxy(is,j)=0
c
      do 10 j=1,4
   10 axy(is,j)=0.
c
      if (ntrunc(is).le.0) go to 80
      np1=keybl(is,1)
      if (np1.eq.0) go to 70
      np2=keybl(is,2)
c
   15 do 20 j=1,2
      xy(1,j)=bline(np1,j)
      xy(2,j)=bline(np2,j)
      nxy(is,j)=0
   20 continue
c
c *** Call subroutine ENDPTS to set inputs for and call
c *** SIMULC to solve the simultaneous equations of the
c *** boundary line and the fracture circle.
c *** Store the points in rows 3 and 4 of xy.
c
      call ENDPTS (flcoef,is,dk,rf,nroots,xy)
c
      go to (25,30,35,40),ntrunc(is)
c
   25 go to (75,50,26),nroots+1
   26 ntrunc(is)=2
      go to 50
c
   30 go to (75,31,50),nroots+1
   31 ntrunc(is)=1
      go to 50
c
```

```
   35 go to (36,36,50),nroots+1
   36 nflag(ifold)=-1
      go to 100
c
   40 go to (36,31,50),nroots+1
c
c
c *** Call subroutine COMSEG to determine the endpoints of
c *** the boundary node.
c
   50 do 55 k=1,4
   55 np(k)=k
      call COMSEG(xy,4,np,nnpt,2,ierr)
      if (ierr.eq.0) go to 57
      write(2,120) ifrac,is
      stop
c
c     if boundary node is a point node and adjacent
c     to another boundary node, discard it.
c
c
   57 if (nnpt.eq.0) go to 70
      if (nnpt.eq.1)then
        do k=1,4
          if (np(k).eq.1.or.np(k).eq.2)go to 70
        enddo
       endif
c
c *** Store the boundary node in binte and axy(6,4).
c
      nbn=nbn+1
      kbn=kbn+1
      if(kbn.gt.maxint)then
        write(2,130)kbn,maxint,ifrac
        stop
       endif

c
c     if fracture was not previously known as a boundary fracture,
c     permute it with first non boundary fracture.
c
      if(ifrac.gt.nfracb)then
        nfracb=nfracb+1
        iref(ifrac)=iref(nfracb)
        iref(nfracb)=ifold
        ifrac=nfracb
       endif
c
c
      ibside(1,kbn)=is
      ibside(2,kbn)=is
      int=1
      k1=0
c
      do 65 k=1,4
      if (np(k).eq.0) go to 65
      knpt=np(k)
      binte(int,kbn)  =xy(knpt,1)
      binte(int+1,kbn)=xy(knpt,2)
      binte(int+2,kbn)=zplane
```

```
      int=int+3
c
      k1=k1+1
      axy(is,2*k1-1)=xy(knpt,1)
      axy(is,2*k1)=xy(knpt,2)
      nxy(is,k1)=1
      if(knpt.ge.3) nxy(is,k1)=2
c
   65 continue
c
c     set orientation of segment to record where is the
c     inside of the flow region.
c
      ksidp=2-kside
      ispar=iside(kpair,ksidp)
      a=flcoef(is,1)*(binte(5,kbn)-binte(2,kbn))
      b=flcoef(is,2)*(binte(4,kbn)-binte(1,kbn))
      c=(flcoef(ispar,3)-flcoef(is,3))*(a-b)
      if(c.gt.0)then
        a=binte(1,kbn)
        b=binte(2,kbn)
        binte(1,kbn)=binte(4,kbn)
        binte(2,kbn)=binte(5,kbn)
        binte(4,kbn)=a
        binte(5,kbn)=b
       endif
c
      ibinte(1,kbn)=ifold
      ibinte(2,kbn)=-is
      go to 80
c
   70 ntrunc(is)=-2
      go to 77
   75 ntrunc(is)=0
   77 ntrunc(7)=ntrunc(7)-1
c
   80 continue
c
      ibkey(ifold,1)=nbn
      ibkey(ifold,2)=kib
      kib=kib+nbn
c
      if (nbn.eq.0) nflag(ifold)=-1
c
c
  100 return
c
  120 format (1h0,' Error sub. bndnod, ierr=1 from comseg.',
     1        /,' fracture',i4,' side',i2)
  130 format (1h0,' Error sub. bndnod, number of boundary nodes',
     1  i6,/,1x,' exceeds maximum dimension, maxint=',i6,/,
     2  1x,' This occurs at fracture number ',i4)
c
      end
      SUBROUTINE BNDPTS(flcoef,ln1,ln2,nf,x,y,npara)
c
      dimension flcoef(6,4)
c
c *** Set up parameters and call SIMULL to determine the
c *** point of intersection, (x,y), of lines ln1 and ln2,
```

```
c *** where ai*x + bi*y + ci = 0.
c
      a1= flcoef(ln1,1)
      b1= flcoef(ln1,2)
      c1= flcoef(ln1,3)
      a2= flcoef(ln2,1)
      b2= flcoef(ln2,2)
      c2= flcoef(ln2,3)
c
      call SIMULL(a1,b1,c1,a2,b2,c2,x,y,npara)
c
      return
c
      end
      SUBROUTINE BNDSEG(iside,bline,keybl,nbp,ierr)
c
c *** Given the 12 or fewer points of intersection of the
c *** 6 boundary lines, determine the 6 or fewer points,
c *** 2 or fewer per line, which define the line segments
c *** of the flow region boundary in the fracture plane.
c
      dimension iside(3,2), bline(12,2),keybl(6,6)
      dimension npt4(4)
c
c *** If one pair of boundary planes is parallel to the
c *** fracture plane then there are only four boundary
c *** lines in that plane and four points of intersection.
      if (nbp.eq.4) go to 50
c
      nbp=0
   10 do 40 k1=1,3
      do 40 k2=1,2
      ln1=iside(k1,k2)
c
      n=0
      do 25 j1=1,3
      if (j1.eq.k1) go to 25
      do 20 j2=1,2
      ln2=iside(j1,j2)
      n=n+1
      npt4(n)=keybl(ln1,ln2)
   20 continue
   25 continue
c
      call COMSEG(bline,12,npt4,nn,1,ierr)
      if (ierr.eq.0) go to 27
      write(2,110) ln1,ln2
      go to 100
c
c *** If a line segment on side ln1 forms a flow region
c *** boundary, (nn=2), store the key to the endpoints
c *** in the first two columns of keybl.  Enter zeros to
c *** indicate intersections outside the flow region.
c
   27 n=0
      if (nn.lt.2) go to 32
      nbp=nbp+1
c
      do 30 i=1,4
      if (npt4(i).eq.0) go to 30
```

```
      n=n+1
      keybl(ln1,n)=npt4(i)
   30 continue
c
   32 do 35 i=n+1,6
      keybl(ln1,i)=0
   35 continue
c
   40 continue
      go to 100
c
   50 do 60 i=1,6
      n=1
      do 55 j=1,6
      if (keybl(i,j).eq.0) go to 55
      if (n.eq.j) go to 53
      keybl(i,n)=keybl(i,j)
      keybl(i,j)=0
   53 n=n+1
   55 continue
   60 continue
c
  100 return
c
  110 format(1h0,' called by bndseg, sides ',2i3)
c
      end
      SUBROUTINE CIRCPT(bxy,j,dk,rf,ierr)
c
c *** Given a point, (bxy(j,1),bxy(j,2)), inside a circle
c *** with center (dk,0) and radius rf, find the point where
c *** the line from the center through the point intersects
c *** the circle and store that point in (bxy(j+2,1),bxy(j+2,2)).
c
      dimension bxy(4,2)
c
      ierr=0 .
c
      xm=bxy(j,1)
      ym=bxy(j,2)
      di=sqrt((xm-dk)*(xm-dk)+ym*ym)
      if(di.le.0.001)then
        ierr=1
        write(2,100)
  100   format('point at center of fracture')
       else
        bxy(j+2,1)=dk+(xm-dk)*rf/di
        bxy(j+2,2)=ym*rf/di
       endif
      return
c
      end
      SUBROUTINE COMPT (xy,n,np1,np2,np3,loc)
c
c *** This subroutine determines whether np3 is between
c *** np1 and np2, coincident with one of them or not between.
c *** It returns 'loc' which equals zero if outside, 1 if coin-
c *** cident with np1, 2 if coincident with np2, and 3 if in
c *** between the two.
c
```

```
      dimension xy(n,2)
      real*8 d1,d2,d3,diff
      real*8 xyd(3,2)
c
      xyd(1,1)=xy(np1,1)
      xyd(1,2)=xy(np1,2)
      xyd(2,1)=xy(np2,1)
      xyd(2,2)=xy(np2,2)
      xyd(3,1)=xy(np3,1)
      xyd(3,2)=xy(np3,2)
      d1=sqrt((xyd(1,1)-xyd(2,1))**2+(xyd(1,2)-xyd(2,2))**2)
      d2=sqrt((xyd(1,1)-xyd(3,1))**2+(xyd(1,2)-xyd(3,2))**2)
      d3=sqrt((xyd(2,1)-xyd(3,1))**2+(xyd(2,2)-xyd(3,2))**2)
c
      diff=abs(d2+d3-d1)
      if (diff .gt. .0005) go to 20
c
      loc=3
      if (d2 .lt. 0.00005) loc=1
      if (d3 .lt. 0.00005) loc=2
      go to 30
c
   20 loc=0
      if (d3 .lt. d2) loc=-1
c
   30 return
c
      end
      SUBROUTINE COMSEG(xy,n,np,nn,iopt,ierr)
c
c *** Given four points in the xy array which define two
c *** line segments, keyed by np(4), determine which points,
c *** if any, define a common line segment and adjust the np
c *** array to key the two or fewer points of that segment.
c
      dimension xy(n,2),np(4)
      real*8 d(6),xyd(4,2)
      real*8 dismin/0.005/
      real*8 difmin/0.005/
      logical p1,p2,p3,p4
      ierr=0
c
c *** Compute the six distances between the four points
c *** rounded off to 4 decimal places:
c *** d(1)=p1 to p2; d(2)=p1 to p3; d(3)=p1 to p4;
c *** d(4)=p2 to p3; d(5)=p2 to p4; d(6)=p3 to p4.
c
      do 5 i=1,4
      ip=np(i)
      do 5 j=1,2
      xyd(i,j)=xy(ip,j)
    5 continue
c
      k=0
      do 10 i=1,3
      do 10 j=i+1,4
      k=k+1
      d(k)=sqrt((xyd(i,1)-xyd(j,1))**2+(xyd(i,2)-xyd(j,2))**2)
   10 continue
c
```

```
      p1=.false.
      if (abs(d(2)+d(3)-d(6)) .lt. difmin) p1=.true.
      p2=.false.
      if (abs(d(4)+d(5)-d(6)) .lt.  difmin) p2=.true.
      p3=.false.
      if (abs(d(2)+d(4)-d(1)) .lt. difmin) p3=.true.
      p4=.false.
      if (abs(d(3)+d(5)-d(1)) .lt. difmin) p4=.true.
c
      if (p3.and.p4) go to 20
      if (p1.and.p2) go to 30
      if (p3.eqv.p4) go to 40
      if (p3.and.p1) go to 50
      if (p3.and.p2) go to 60
      if (p4.and.p1) go to 70
      if (p4.and.p2) go to 80
      write(2,110)
      ierr=1
      go to 100
c
   20 np(1)=0
      np(2)=0
      nn=2
      if (d(6) .gt. dismin) go to 100
      if ((p1 .or. p2) .and. (iopt .ne. 3)) go to 42
      nn=1
      go to 100
c
   30 np(3)=0
      np(4)=0
      nn=2
      go to 100
c
   40 np(1)=0
      np(2)=0
   42 np(3)=0
      np(4)=0
      nn=0
      go to 100
c
   50 np(2)=0
      np(4)=0
      nn=2
      if (d(2) .gt. dismin) go to 100
      go to (40,40,75),iopt
c
   60 np(1)=0
      np(4)=0
      nn=2
      if (d(4) .gt. dismin) go to 100
      go to (40,40,85),iopt
c
   70 np(2)=0
      np(3)=0
      nn=2
      if (d(3) .gt. dismin) go to 100
      go to (40,40,75),iopt
   75 np(1)=0
      nn=1
      go to 100
```

```
c
   80 np(1)=0
      np(3)=0
      nn=2
      if (d(5) .gt. dismin) go to 100
      go to (40,40,85),iopt
   85 np(2)=0
      nn=1
c
  100 return
c
  110 format (1h0,' error in subroutine comseg')
c
      end
      SUBROUTINE DISCAR (nfrac)
c
c *** This subroutine eliminates fractures lying outside the flow
c *** region.  These fractures, flagged for discarding by
c *** subroutine flag or intfra, are marked nflag(n)=-1.
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      include 'comset.blkf/list'
      include 'comfint.blkf/list'
      include 'comout.blkf/list'
      include 'comref.blkf/list'
c
      k=1
c
      do l=1,nsets
        iseti(l,2)=0
       enddo
c
      do ifr=1,nfrac
c
        ifold=iref(ifr)
        if (nflag(ifold).ge.0)then
          iref(k)=ifold
          inew(ifold)=k
c
          k=k+1
          l=isetfr(ifold)
          iseti(l,2)=iseti(l,2)+1
         else
          inew(ifold)=0
         endif
       enddo
      nfrac=k-1
c
      return
c
      end
      SUBROUTINE DISCOUT(nfrac)
c
c     among the intersections outside the flow region, which are
c     recorded only for plotting purposes, discard the ones which
c     lay on fracture flagged for discarding.  Use common nodes
c     for temporary arrays to save space.
c
      include 'maxdim.blkf/list'
```

```
      include 'comfrc.blkf/list'
      include 'comout.blkf/list'
      include 'comref.blkf/list'
      include 'comsubi.blkf/list'
      common /nodes/ ilaso(maxcub),ifiro(maxcub),nexto(maxout*3),
     '               ninto(maxout*3)
c
      do i=1,ncub
        ifiro(i)=0
       enddo
      do i=1,maxout*3
        nexto(i)=0
       enddo
      nnn=0
      nentry=0
      do iout=1,nout
        ifold1=ifout(iout,1)
        ifold2=ifout(iout,2)
        if (nflag(ifold1).ge.0.and.nflag(ifold2).ge.0)then
          nnn=nnn+1
          ifout(nnn,1)=ifold1
          ifout(nnn,2)=ifold2
          do i=1,6
            endout(i,nnn)=endout(i,iout)
           enddo
          do ientc=icubk(ifold1),icubk(ifold1+1)-1
            ic=icub(ientc)
            nentry=nentry+1
            if(ifiro(ic).eq.0)then
              ifiro(ic)=nentry
             else
              nl=ilaso(ic)
              nexto(nl)=nentry
             endif
            ilaso(ic)=nentry
            ninto(nentry)=nnn
           enddo
          do ientc=icubk(ifold2),icubk(ifold2+1)-1
            ic=icub(ientc)
            if(ifiro(ic).ne.0)then
              nl=ilaso(ic)
              if(ninto(nl).eq.nnn)goto 100
              nexto(nl)=nentry+1
             else
              ifiro(ic)=nentry+1
             endif
            nentry=nentry+1
            ilaso(ic)=nentry
            ninto(nentry)=nnn
100         continue
           enddo
         endif
       enddo
      nout=nnn
      ientry=1
      do ic=1,ncub
        ifrok(ic)=ientry
        nentry=ifiro(ic)
        do while(nentry.ne.0)
          ifro(ientry)=ninto(nentry)
```

```
          nentry=nexto(nentry)
          ientry=ientry+1
         enddo
       enddo
      if(ientry.gt.maxout*3)then
        write(2,10) ientry,maxout*3
   10   format(' in subroutine discout, '/' ientry= ',i6/
     ,                                    ' maxim.= ',i6)
        stop
       endif
      ifrok(ncub+1)=ientry
      return
c
      end
      SUBROUTINE ENDPTS(flcoef,is,dk,rf,nroots,xy)
c
c *** Set up inputs and call simulc to solve the simultaneous
c *** equations of line is and circle nf.  Store the resulting
c *** points if rows 3 and 4 of xy(4,2).
c
      dimension flcoef(6,4),xy(4,2)
c
      a=flcoef(is,1)
      b=flcoef(is,2)
      c=flcoef(is,3)
c
      call SIMULC(a,b,c,dk,0.0,rf,nroots,x1,y1,x2,y2)
c
      xy(3,1)=x1
      xy(3,2)=y1
      xy(4,1)=x2
      xy(4,2)=y2
c
      return
c
      end
      SUBROUTINE EQPLAN (nfrac)
c
c
c *** This subroutine calculates the coefficients ,a,b,c and d, of the
c *** plane on which the fracture lies given the orientation angles phi
c *** and theta and the coordinates of the center of the plane,(xc,yc,zc).
c
c *** nfrac is the number of fractures in the generation region.
c
c *** The components of frac used in the calculations are:
c          frac(1,i) = phi
c          frac(2,i) = theta
c          frac(5,i) = xc
c          frac(6,i) = yc
c          frac(7,i) = zc.
c
c *** The general form of a plane is ax+by+cz+d=0.
c *** The coefficients are stored in the array frac:
c          frac(9,i) = a
c          frac(10,i) = b
c          frac(11,i) = c
c          frac(12,i) = d.
c
c
```

```
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      real lambda,mu,nu
      parameter (pi180=0.0174532925)
c
c
      do 10 i=1,nfrac
      phi = frac(1,i)*pi180
      theta = frac(2,i)*pi180
      siphi = sin(phi)
      lambda = siphi*cos(theta)
      mu = -siphi*sin(theta)
      nu = cos(phi)
c
      frac(9,i) = lambda
      frac(10,i) = mu
      frac(11,i) = nu
      frac(12,i) = -(lambda*frac(5,i)+mu*frac(6,i)+nu*frac(7,i))
   10 continue
c
      return
c
      end
      SUBROUTINE EXPDIS (k,n1,n2,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
c *** y = -ln(1-x)/lambda is distributed exponentially with
c *** parameter lambda.  The expected value of y is 1/lambda.
c
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
c
      real lambda
      real*8 dseed
c
      lambda=1./ev
c
      do 10 I=n1,n2
   10 frac(k,i)=-alog(1.-GGUBFS(dseed))/lambda
c
      return
c
      end
      SUBROUTINE FLAG (flcoef,zplane,dk,rf,iside,
     1                 iflag,ntrunc,itole)
c
c *** This subroutine compares the position of the given fracture
c *** with the lines of intersection of the flow region boundary
c *** planes with the fracture plane.  The coefficients of the
c *** boundary lines are stored in flcoef.  If the fracture lies outside
c *** the flow region it is flagged for discarding.  If the fracture
c *** lies partially outside the flow region it is flagged for
c *** truncation at one or more boundary planes.
c
      dimension flcoef(6,4),iside(3,2)
      dimension ntrunc(7)
```

```
c
c *** Working with parallel pairs of boundary lines, designated
c *** l1 and l2, set iflag and ntrunc(i),i=1,7 accordingly.
c *** If the fracture plane is parallel to any pair of boundary
c *** planes, go to 80 to determine the relative positions of the
c *** planes.
c
      do 100 k=1,3
c
      l1=iside(k,1)
      l2=iside(k,2)
c
c *** Are the boundary planes parallel to the fracture plane?
      a1=flcoef(l1,1)
      b1=flcoef(l1,2)
      if (iround(a1,itole).eq.0.and.iround(b1,itole).eq.0) go to 80
c
c *** In the plane z=zplane, does the fracture with center (dk,0)
c *** and radius rf lie between the lines l1 and l2?
c *** Compute the perpendicular distances d1 {between (dk,0) and l1},
c *** d2 {between (dk,0) and l2}, and d12 {between l1 and l2}.
c *** Compare distances rounded off to itole places.
c
      sra2b2=sqrt(flcoef(l1,1)**2+flcoef(l1,2)**2)
      d1=abs(flcoef(l1,1)*dk+flcoef(l1,3))/sra2b2
      id1=iround(d1,4)
      d2=abs(flcoef(l2,1)*dk+flcoef(l2,3))/sra2b2
      id2=iround(d2,4)
      d12=abs(flcoef(l2,3)-flcoef(l1,3))/sra2b2
      diff=d1+d2-d12
      idiff=iround(diff,4)
      irf=iround(rf,4)
c
c *** Save fracture as is or discard?
c
      if (idiff.gt.0) go to 50
      if (id1.gt.irf .and. id2.gt.irf) go to 100
      go to 60
c
   50 if (id1.ge.irf .and. id2.ge.irf) go to 110
c
c *** Fracture intersects l1 and/or l2, determine truncation
c *** codes for each line.
c
   60 iflag=1
c
c *** Call subroutine FLAGNT to set codes for side l1.
      call FLAGNT(ntrunc,l1,id1,irf,id2,idiff)
c
c *** Call subroutine FLAGNT to set codes for side l2.
      call FLAGNT(ntrunc,l2,id2,irf,id1,idiff)
c
      go to 100
c
c *** Fracture plane is parallel to boundary planes.
c
   80 if (iround(zplane,itole).eq.0) go to 85
      plane1=-flcoef(l1,4)*zplane/(flcoef(l1,3)-flcoef(l1,4))
      plane2=-flcoef(l2,4)*zplane/(flcoef(l1,3)-flcoef(l1,4))
      d1=abs(plane1-zplane)
```

```
      id1=iround(d1,itole)
      d2=abs(plane2-zplane)
      id2=iround(d2,itole)
c
c *** Discard if fracture plane coincides with boundary plane.
      if (id1.eq.0 .or. id2.eq.0) go to 110
c
c *** Discard if fracture plane is not between plane1 and plane2.
      d12=abs(plane1-plane2)
      id12=iround(d12,itole)
c
      if (id1+id2.gt.id12) go to 110
c
c *** Set flags, fracture plane parallel to and between
c *** plane1 and plane2.
   85 ntrunc(l1)=-1
      ntrunc(l2)=-1
c
  100 continue
c
      return
c
c *** Flag fracture for discarding
  110 iflag=-1
      return
c
      end
      SUBROUTINE FLAGNT(ntrunc,ln,idn,irf,idm,idiff)
c
c *** Determine the truncation code for side ln which
c *** is a perpendicular distance dn from the fracture
c *** center.  The corresponding distance to the parallel
c *** boundary line is dm and the difference between dn+dm
c *** and the distance between the two lines is diff, which
c *** equals zero if the fracture center lies between the
c *** two lines.
c
      dimension ntrunc(7)
c
      if (idn.gt.irf) go to 60
      if (idn.ne.0) go to 20
      ntrunc(ln)=4
      go to 50
c
   20 if (idn.ne.irf) go to 40
      ntrunc(ln)=1
      go to 50
c
   40 ntrunc(ln)=2
      if (idiff.gt.0 .and. idm.gt.idn) ntrunc(ln)=3
c
   50 ntrunc(7)=ntrunc(7)+1
c
   60 return
c
      end
      SUBROUTINE FRAGEN (nfrac,itole,iranf,dseed,title,rgene)
c
c     ****************************************************************
c
```

```
c *** This subroutine reads in fracture information and then either
c *** reads in or generates the following fracture characteristics -
c *** orientation, radius, 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       rgene = radius of spherical generation region
c
c       frac(1,i) = orientation angle phi
c       frac(2,i) = orientation angle theta
c       frac(3,i) = radius
c       frac(4,i) = aperture
c       frac(5,i) = xc, x-coordinate of fracture center
c       frac(6,i) = yc, y-coordinate of fracture center
c       frac(7,i) = zc, z-coordinate of fracture center
c       frac(8,i) = area, adjusted if fracture is truncated
c
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
c       ichar = 1 -- read values for each orientation, radius,
c                    and aperture
c             = 2 -- set characteristic equal to a constant value
c             = 3 -- generate values of characteristic according
c                    to a distribution
c
c       idist = 1 -- normal distribution
c             = 2 -- lognormal diatribution
c             = 3 -- exponential distribution
c             = 4 -- normal distribution correlating aperture
c               with radius
c             = 5 -- uniform distribution
c               Other distribution options will be added as needed
c
c *** In addition, the following variables are read in --
c
c       nfrac = number of fractures per set
c
c       rlamb = density of fractures, i.e. fractures/unit volume
c
c       itole = number of decimal places in fracture center coordinates
c
c       const = constant value, see ichar = 2
c
```

```
c       ev = expected value of statistical distribution, or slope
c            for idist = 4, or lower limit, amin, for idist = 5
c
c       sd = standard deviation of statistical distribution, or
c            upper limit, amax, for idist = 5
c
c       ycept = parameter for idist = 4
c
c *** The array iseti is used to store the values of nfrac, icent,
c *** ichar and idist for each set.  The array rseti is used to store
c *** the values of const, ev, and sd for each fracture set.
c
c     ******************************************************************
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      include 'comset.blkf/list'
      include 'comref.blkf/list'
      integer title(40)
      parameter (pi=3.1415927)
c
      real*8 dseed
c
      read (1,280) iranf,dseed
c *** If iranf = 0, call GGUBFS (random number generator)
c *** If iranf = 1, read dseed and call GGUBFS
      if(iranf .eq. 0) dseed = secnds(0.0) * 100.0
c
c *** Read title and input variables
c
      read (1,270) (title(i),i=1,40)
      read (1,260) rgene
      read (1,240) nsets,itole
c
c *** Zero information matrices
c
      do i=1,nsets
        do j=1,8
          iseti(i,j)=0
          rseti(i,j)=0.
          rseti(i,j+8)=0.
         enddo
       enddo
c
      genvol=(4./3.)*pi*rgene**3
      write(2,320) title,rgene,genvol
      write(2,290) dseed
      n1=1
c
      do i=1,nsets
c
        read (1,240) icent,idens
        iseti(i,1)=icent
c
        if(icent.eq.1)then
c
c         Read in all fracture characteristics
c
          read (1,210) nfrac
          iseti(i,2)=nfrac
```

```
            n2=nfrac+n1-1
            read (1,250) ((frac(j,m),j=1,7),m=n1,n2)
            iseti(i,3)=1
            iseti(i,5)=1
            iseti(i,7)=1
            rlamb=float(nfrac)/genvol
            rseti(i,13)=rlamb
c
c           Call FRASTA to compute original mean and stand.
c           deviation for each parameter
c
            do jj=1,4
              j=jj
              call FRASTA (j,n1,n2,ev,sd)
              m=3*j-1
              rseti(i,m)=ev
              rseti(i,m+1)=sd
             enddo
c
           else
c
            Generate locations of fracture centers
c
            if(idens.eq.0)then
              read (1,260) rlamb
              nfrac=int(genvol*rlamb)
             else
              read (1,210) nfrac
              rlamb= float(nfrac)/genvol
             endif
c
            n2=nfrac+n1-1
            iseti(i,2)=nfrac
            rseti(i,13)=rlamb
            if(n2.gt.maxfrc)then
              write(2,300) n2,maxfrc
              write(2,310) i
              stop
             endif
c
            call RANCEN (n1,n2,dseed,rgene,itole)
c
            k1=0
            l=1
            m=-2
c
            do k=1,3
              l=l+2
c
              read (1,210) ichar
              iseti(i,l)=ichar
c
              if(ichar.ne.3)then
c
c               Set fracture characteristic to a constant
c
                do while(k1.le.k)
                  m=m+3
                  k1=k1+1
                  read (1,260) const
```

```
                rseti(i,m)=const
                do j=n1,n2
                  frac(k1,j)=const
                 enddo
                if (k.eq.2)then
                  area=pi*const*const
                  rseti(i,14)=area
                  do j=n1,n2
                    frac(8,j)=area
                   enddo
                 endif
               enddo
             else
c
c             Read in code for statistical distribution
c             and read in statistical parameters.  Call
c             appropriate distribution function subroutine
c             to generate fracture characteristics.
c
              do while(k1.le.k)
                m=m+3
                k1=k1+1
                read (1,230) idist,ev,sd,ycept
                iseti(i,l+1)=idist
                rseti(i,m)=ycept
                rseti(i,m+1)=ev
                rseti(i,m+2)=sd
                if(idist.eq.1)then
                  CALL NORDIS (k1,n1,n2,dseed,ev,sd)
                 elseif(idist.eq.2)then
                  CALL LOGDIS (k1,n1,n2,dseed,ev,sd)
                 elseif(idist.eq.3)then
                  CALL EXPDIS (k1,n1,n2,dseed,ev)
                 elseif(idist.eq.4)then
                  CALL NORDIS1(k1,n1,n2,dseed,ycept,ev,sd)
                 elseif(idist.eq.5)then
                  CALL UNIDIS (k1,n1,n2,dseed,ev,sd)
                 endif
               enddo
             endif
           enddo
         endif
c
c       Compute fracture areas
c
        if(rseti(i,14).eq.0..or.icent.eq.1)then
          do j=n1,n2
            frac(8,j)=pi*frac(3,j)**2
           enddo
          call FRASTA (8,n1,n2,ev,sd)
          rseti(i,15)=ev
          rseti(i,16)=sd
         endif
c
c       set up array of set numbers and "clean" angles
c
        do j=n1,n2
          isetfr(j)=i
          if(frac(1,j).lt.0.)then
            frac(1,j)=-frac(1,j)
```

```
            frac(2,j)=frac(2,j)+180.
           elseif(frac(1,j).gt.90.)then
            frac(1,j)=180.-frac(1,j)
            frac(2,j)=frac(2,j)+180.
           endif
          if(frac(2,j).gt.360.)frac(2,j)=frac(2,j)-360.
         enddo
c
c       end loop on sets
c
        n1=n2+1
       enddo
      nfrac=n2
c
c     Program stop if nfrac is greater than maxfrc.
c
      if(nfrac.gt.maxfrc)then
        write(2,300) nfrac,maxfrc
        stop
       endif
c
      return
c
  210 format(10X,I5)
  230 format(10X,I5,3(5x,f10.4))
  240 format(2(10X,I5))
  250 format(7F10.4)
  260 format(10X,F10.4)
  270 format(20a4)
  280 format(10x,i5,15x,d15.8)
  290 format(1h0,42hthe initial seed used in the random number,
     '       14h generator is ,d15.8)
  300 format (1h0,40h----------------------------------------,/,
     ' 50hprogram stop, nfrac is greater that maxfrc, nfrac=,i5,/,
     ' 7hmaxfrc=,i5,/,1x,40h----------------------------------------)
  310 format(1h0,39hbefore calling rancen for fracture set ,i3)
  320 format(1h0,2(20a4,/,1x),20hspherical generation,
     ' 16h region, radius=,f9.2,12h and volume=,e15.5)
c
      end
      SUBROUTINE FRASTA (k,n1,n2,ev,sd)
c
c *** This subroutine calculates the mean and standard deviation.
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
c
      n=0
      s1=0.
      s2=0.
c
      do i=n1,n2
        n=n+1
        c=frac(k,i)
        s1=s1+c
        s2=s2+c*c
       enddo
c
      d=float(n)
      ev=s1/d
```

```
      sd=sqrt(abs(s2/d-ev*ev))

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

c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      include 'comfint.blkf/list'
      include 'comout.blkf/list'
      include 'comref.blkf/list'
      include 'comsubi.blkf/list'
      common /nodes/ifirsi(maxfrc),ilasi(maxfrc),nexti(maxint)
      dimension frcoef(6,4),flcoef(6,4),iside(3,2),rotf(3,3)
      dimension xy(4,2),nside(2)
c     equivalence (binte,inte1),(ibside,inside)
c
c
c     initialize level one
c
      iprev2=0
      iprev3=0
      ilevel=0
      n2=0
      n3=nfracb
      nint=0
      kin=2
      inst=ninb
      nout=0
      nnp=0
      do i=1,maxfrc
        ifirsi(i)=0
        ilasi(i)=0
       enddo
      do i=1,2*maxint
        nexti(i)=0
       enddo
c
c     begin loop on levels
c
      do while (n3.gt.n2)
        ilevel=ilevel+1
        n1=n2+1
        n2=n3
        iprev1=iprev2+1
        iprev2=iprev3
```

```
c
c       begin loop on fractures in level # ilevel
c
        do ifrac=n1,n2
          ifold=iref(ifrac)
c
          rf=frac(3,ifold)
          nbn=ibkey(ifold,1)
          nif=0
          nside(1)=0
          nside(2)=0
c
c         look for previously stored nodes
c
          noldi=0
          if (iprev3.ne.0)then
            nni=ifirsi(ifold)
            do while(nni.ne.0)
              ni=inst+nni
              inte3(kin+nif)=nni
              nif=nif+1
              noldi=noldi+1
              nni=nexti(nni)
            enddo
          endif
c
c         Rotate fracture and flow region boundaries to a system
c         convenient to fracture plane ifrac.
c
          call ROFRAC(ifold,rotf,zplane,dk)
c
c         Find lines of intersection between boundary planes
c         and plane ifrac, if fracture intersects boundary.
c
          if (ilevel.eq.1)call RFLOFRC(frcoef,rotf,zplane,flcoef)
c
c         begin loop over fractures not previously encountered.
c
          k1=inst+nint+1
          noutold=nout
c
c         look only at fractures in same cubes as i
c
c         loop over cubes intersected by i
c
          do 180 ientc=icubk(ifold),icubk(ifold+1)-1
          ic=icub(ientc)
c
c         loop over fractures intersecting cube ic
c
          do 180 ientf=ifcuk(ic),ifcuk(ic+1)-1
            jfold=ifcu(ientf)
            jfrac=inew(jfold)
            if(jfrac.le.ifrac)goto 180
            rj=frac(3,jfold)
c
c           Rotate fracture jfrac to coord. system of fracture ifrac and
c           determine the line of intersection between the planes,
c           aj x + bj y + cj = 0.
c
```

```
            call ROJTOF (jfold,rotf,zplane,aj,bj,cj,dj,xj,yj,zj)
c
c           if jfrac centre too far from ifrac plane, no intersection.
c
            dist=abs(zj-zplane)
            if(dist.gt.rj)goto 50
c
c           If planes ifrac and jfrac are parallel, there is no intersection.
c
            ird1=iround(aj,itole)
            ird2=iround(bj,itole)
            if(ird1.eq.0.and.ird2.eq.0)goto 50
c
c           Call subroutine INTFRC to determine the intersection, if any,
c           between fractures ifrac and jfrac.
            call INTFRC (ifrac,dk,zplane,rf,jfrac,aj,bj,cj,
     ,                   xj,yj,zj,rj,itole,xy,ntf)
            if(ntf.eq.1)then
c
c             check if intersection has not already been found.
c
              do ientry=kin,kin+nif-1
                int=inte3(ientry)+inst
                do i=1,2
                  if(inte2(i,int).eq.jfold)goto 50
                 enddo
               enddo
c
c             Truncate fracture/fracture node at boundary nodes.
c             Note: inte1=binte, inside=ibside.
c
              if(ilevel.eq.1)then
                do io=noutold+1,nout
                  do i=1,2
                    if(ifout(io,i).eq.jfold)goto 50
                   enddo
                 enddo
                call TRUNCN (ifold,flcoef,iside,nbn,jfold,
     ,                       aj,bj,cj,xy,nside,nn)
                if(nn.eq.0) go to 50
               endif
c
c             Store fracture/fracture intersection and truncation code.
c
              nint=nint+1
              ni=inst+nint
              iprev3=iprev3+1
              if(ni.ge.maxint)then
                write(2,150) ninb,nint,maxint
                stop
               endif
c
c             increment array nexti to keep track of intersections on
c             fracture jfold.
c
              nnp=nnp+1
              if(ifirsi(jfold).eq.0)then
                ifirsi(jfold)=nnp
               else
                jl=ilasi(jfold)
```

```
              nexti(jl)=nnp
             endif
            ilasi(jfold)=nnp
c
c           if this fracture does not intersect fractures previously
c           treated at this level (i.e. 'new' fracture), increment
c           index and position fracture for next level.
c
            if(jfrac.gt.n3)then
              n3=n3+1
              nfold=iref(n3)
              iref(jfrac)=nfold
              iref(n3)=jfold
              inew(jfold)=n3
              inew(nfold)=jfrac
             endif
c
            k=1
            do i=1,2
              intel(k,ni)  =xy(i+2,1)
              intel(k+1,ni)=xy(i+2,2)
              intel(k+2,ni)=zplane
              inside(i,ni)=nside(i)
              k=4
             enddo
c
            inte2(1,ni)=ifold
            inte2(2,ni)=jfold
            inte3(kin+nif)=nint
            nif=nif+1
c
           endif
   50     continue
  180    continue
         if(nif.ne.noldi)then
c
c         Call RFRCGL0 to rotate new intersections from local
c          to global coordinates.
c
          k2=ni
          call RFRCGL0 (10,maxint+1,intel,k1,k2,rotf,1)
         endif
        inkey(ifold,1)=nif
        inkey(ifold,2)=kin
        kin=kin+nif
c
c       first level: record intersections outside flow region
c       for plotting program, and rotate boundary nodes.
c
        if(ilevel.eq.1)then
          if (nout.gt.noutold)then
            nout1=noutold+1
            do nn=nout1,nout
              endout(3,nn)=zplane
              endout(6,nn)=zplane
             enddo
            call RFRCGL0(6,maxout,endout,nout1,nout,rotf,1)
           endif
c
          k1=ibkey(ifold,2)
```

```
            k2=k1+ibkey(ifold,1)-1
            call RFRCGLO (10,maxint+1,inte1,k1,k2,rotf,1)
           endif
c
         enddo
       enddo
      inte3(1)=kin-1
c
      return
c
c
  150 format (1h0,28hThe number of boundary nodes,i5,5h plus,/,1x,
     ,            28hthe number of internal nodes,i5,/,1x,
     ,      28h exceeds the maximum        ,i5,/,1x,10hError Stop)
c
      end
c
      SUBROUTINE INTFRA (nfrac,ndiscd)

c
c *** Find non-conducting fractures, (those with fewer than two inter-
c *** sections, internal and/or boundary), eliminate them from
c *** intersection arrays, and flag the fractures for discarding.
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      include 'comfint.blkf/list'
      include 'comref.blkf/list'
      common/nodes/newint(maxfrc)
c     equivalence (binte,inte1),(ibside,inside),(ibinte,inte2)
c
c
      ndiscd=0
      inst=ninb
      if1=iref(nfrac+1)
      inkey(if1,2)=inte3(1)+1
c
   15 nelim=0
      do 90 nf=1,nfrac
      nfold=iref(nf)
      if (nflag(nfold) .lt. 0) go to 90
      ink=inkey(nfold,1)
      ibk=ibkey(nfold,1)
      inb=ink+ibk
      if (inb .ge. 2) go to 90
      nflag(nfold)=-1
      ndiscd=ndiscd+1
      if (inb .eq. 0) go to 90
      go to (20,70),ibk+1
c
c *** Eliminate the only intersection on fracture nf, inkey(nf,1)=1
c
   20 nelim=nelim+1
      ient=inkey(nfold,2)-1
      nentr=inkey(nfold,1)
      do 22 i=1,nentr
   21   ient=ient+1
        infj=inte3(ient)
        in=inst+infj
        if(inte2(1,in).eq.0)goto 21
```

```
        if(inte2(1,in).eq.nfold)then
          njold=inte2(2,in)
          goto 23
         elseif(inte2(2,in).eq.nfold)then
          njold=inte2(1,in)
          goto 23
         endif
   22  continue
   23 continue
c
      inte2(1,in)=0
      inkey(njold,1)=inkey(njold,1)-1
      go to 90
c
c *** Eliminate the only boundary node on fracture nf, ibkey(nf,1)=1
c *** Note: inte1=binte, inte2=ibinte, inside=ibside.
c
   70 in=ibkey(nfold,2)
      inte2(1,in)=0
c
   90 continue
c
c *** Recheck for non-conducting fractures if any fracture/fracture
c *** intersections were deleted.  Eliminating one or more may result
c *** in others becoming non-conducting.
      if (nelim .gt. 0) go to 15
c
c     adjust intersections arrays to reflect deletions
c
c     boundary int.
c
      k=0
      do i=1,ninb
        if(inte2(1,i).ne.0)then
          k=k+1
          do l=1,2
            inside(l,k)=inside(l,i)
            inte2(l,k)=inte2(l,i)
          enddo
          do l=1,6
            inte1(l,k)=inte1(l,i)
          enddo
          newint(i)=k
        else
          newint(i)=0
        endif
       enddo
      ninb=k
c
      do i=1,nfrac
        iold=iref(i)
        ib=ibkey(iold,2)
        ibkey(iold,2)=newint(ib)
       enddo
c
c     fracture intersections
c
      k=inst
      do i=inst+1,inst+nint
        if(inte2(1,i).ne.0)then
```

```
            k=k+1
            do l=1,2
              inside(l,k)=inside(l,i)
              inte2(l,k)=inte2(l,i)
             enddo
            do l=1,6
              inte1(l,k)=inte1(l,i)
             enddo
            newint(i)=k-inst
           else
            newint(i)=0
           endif
         enddo
        nint=k
c
        ientry=inte3(1)
        do i=2,ientry
          intold=inte3(i)+inst
          inte3(i)=newint(intold)
         enddo
  100  return
c
       end
       SUBROUTINE INTFRC (nf,dk,zplane,rf,nj,a,b,c,x,y,z,r,
      1                   itole,xy,ntf)
c
       dimension xy(4,2),np(4)
c
c *** Given fracture circle nf with center (dk,0), radius rf,
c *** in zplane and sphere nj with center (x,y,z) and radius r,
c *** determine whether sphere nj intersects plane nf.  If yes,
c *** determine circle or point of intersection, njp, with center
c *** (x,y,zplane) and radius rjp.  If no, there is no
c *** intersection between fractures f and j.
c
c
       rjp2=r*r-(z-zplane)**2
       if (iround(rjp2,itole) .eq. 0) go to 15
       if (rjp2 .lt. 0.0) go to 75
       rjp=sqrt(rjp2)
       go to 20
c
    15 rjp2=0.0
       rjp =0.0
c
c *** Determine the intersection, if any, between circle njp and
c *** line a x + b y + c = 0, this is the intersection of
c *** fracture circle nj with plane nf.
    20 call SIMULC(a,b,c,x,y,rjp,nrootj,x1,y1,x2,y2)
       go to (75,30,40),nrootj+1
c
c *** There is only one point of intersection between circle nj
c *** and plane nf. If this point is inside circle nf store it
c *** in rows 3 and 4 of xy-array.
    30 dist2=(x-dk)**2+y*y
       if (iround((dist2-rf*rf),itole) .gt. 0) go to 75
       xy(3,1)=x1
       xy(3,2)=y1
       xy(4,1)=x1
       xy(4,2)=y1
```

```
      go to 70
c
c *** Determine the intersection, if any, between fracture
c *** circle nf and line a x + b y + c = 0.
   40 call SIMULC(a,b,c,dk,0.0,rf,nrootf,x3,y3,x4,y4)
      if (nrootf .eq. 0) go to 75
c
c *** set up point array for comseg
      xy(1,1)=x1
      xy(1,2)=y1
      xy(2,1)=x2
      xy(2,2)=y2
      xy(3,1)=x3
      xy(3,2)=y3
      xy(4,1)=x4
      xy(4,2)=y4
      go to (42,46),nrootf
c
   42 call COMPT(xy,4,1,2,3,loc)
      if (loc .le. 0) go to 75
      go to 70
c
   46 do 48 i=1,4
   48 np(i)=i
c
      call COMSEG(xy,4,np,nn,3,ierr)
      if (ierr .eq. 0) go to 50
      write(2,150) nf,nj
      stop
c
c *** Store endpoints of intersection in rows 3 and 4 of xy-array.
   50 go to (75,55,60),nn+1
c
   55 np1=np(3)
      if (np1 .ne. 3) np1=4
      np2=7-np1
      xy(np2,1)=xy(np1,1)
      xy(np2,2)=xy(np1,2)
      go to 70
c
   60 if ((np(3)+np(4)) .eq. 7) go to 70
      np1=4
      if (np(3) .ne. 3) np1=3
      np2=2-np(1)
      xy(np1,1)=xy(np2,1)
      xy(np1,2)=xy(np2,2)
      if ((np(1)+np(2)) .ne. 3) go to 70
      xy(4,1)=xy(2,1)
      xy(4,2)=xy(2,2)
c
   70 ntf=1
      return
c
   75 ntf=2
      return
c
  150 format (1h0,32herror in comseg called by intfrc,/,
     1        1x,9hfractures,i5,4h and,i5)
c
      end
```

```
      SUBROUTINE LOGDIS (k,n1,n2,dseed,ev,sd)
c
c *** This subroutine generates random variables distributed
c *** lognormally with expected value ev and standard deviation
c *** sd.
c
c *** Step 1.  sn = sum of 25 random variables distributed
c ***               uniformly in (0,1)
c *** Step 2.  sn = (sn-12.5)/sqrt(25/12) is distributed
c ***               normally with mean = 0 and standard deviation = 1
c *** Step 3.  exp(sd*sn+ev) is distributed lognormally with
c ***          mean = exp(ev)*exp(sd*sd/2) and standard
c ***          deviation = exp(sd*sd+2*ev)*(exp(sd*sd)-1)
c
c
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      parameter (c=0.692820323)
c *** where c=1./sqrt(25./12.)
c
      a=alog(ev*ev+sd*sd)
      b=alog(ev)
c
      evn=2.*b-0.5*a
      sdn=sqrt(a-2.*b)
c
      do 120 i=n1,n2
      sn=0.
      do 110 j=1,25
  110 sn=sn+GGUBFS(dseed)
      sn=c*(sn-12.5)
  120 frac(k,i)=exp(sdn*sn+evn)
c
      return
c
      end
      SUBROUTINE NODELE (nfrac,nseg,nsegb)
c
c *** This subroutine makes up the nodal arrays from the boundary node
c *** and fracture intersection arrays.  If the input value nseg is
c *** greater than one, each node is divided into nseg segments by
c *** subroutine nodseg.  Each boundary node will be divided
c *** into nsegb segments by subroutine nodseg.
c
      include 'maxdim.blkf/list'
      include 'comfint.blkf/list'
      include 'comnod.blkf/list'
      include 'comref.blkf/list'
c     equivalence (inte1,binte),(inside,ibside),(inte2,ibinte)
c
      kin=2
      n1=1
      n2=0
      ilast=0
c
      do 80 nf=1,nfrac
      nfold=iref(nf)
      nkey(nf,2)=kin
      nnf=0
```

```
c
c *** Boundary nodes
c
      nbn=ibkey(nfold,1)
      if (nbn .eq. 0) go to 45
      k1=ibkey(nfold,2)
      k2=k1+nbn-1
      do 40 nbb=k1,k2
      nb=nbb
c
c *** Call sub. NODSEG to divide the node into nsegb pieces and store them
c *** in array node1, starting in location n1.  Store the corresponding
c *** side codes in nside.  Note: inte1=binte, inte2=ibinte,
c *** inside=ibside.
      call NODSEG (nb,nsegb,n1,n2,1)
c
      do 30 nn=n1,n2
      int=inte2(1,nb)
      node2(nn,1)=inew(int)
      int=inte2(2,nb)
      node2(nn,2)=inew(int)
      nnf=nnf+1
      node3(kin)=nn
      kin=kin+1
   30 continue
      n1=n2+1
      if (n1 .gt. maxnod) go to 90
   40 continue
c
c
c *** Fracture Intersections
c
   45 nif=inkey(nfold,1)
      if(nif.eq.0)goto 75
      k=inkey(nfold,2)
      icount=1
      nfold1=iref(nf+1)
      iint=inkey(nfold,1)
      do while(icount.le.iint)
        nin=inte3(k)
        if(nin.eq.0)goto 70
        icount=icount+1
        ni=inst+nin
        if (nin .le. ilast) go to 60
        ilast=nin
c
c *** Call sub. NODSEG to divide the intersection into nseg pieces
c *** and store them in array node1, starting in location n1.  Store
c *** the corresponding side codes in nside.
        call NODSEG (ni,nseg,n1,n2,2)
c
        do 50 nn=n1,n2
        int=inte2(1,ni)
        node2(nn,1)=inew(int)
        int=inte2(2,ni)
        node2(nn,2)=inew(int)
        nnf=nnf+1
        node3(kin)=nn
        kin=kin+1
   50   continue
```

```
        n1=n2+1
        if (n1 .gt. maxnod) go to 90
        go to 70
c
c *** Intersection nin between fractures nf and nj has already been
c *** divided and stored.  Find the nodes and enter them in node3.
   60   njold=inte2(1,ni)
        nj=inew(njold)
        j1=nkey(nj,2)
        j2=j1+nkey(nj,1)-1
c
        do 65 j=j1,j2
        nn=node3(j)
        if (node2(nn,2) .ne. nf) go to 65
        nnf=nnf+1
        node3(kin)=nn
        kin=kin+1
   65   continue
c
   70   k=k+1
       enddo
c
   75 nkey(nf,1)=nnf
c
   80 continue
c
      nnode=n2
      node3(1)=kin-1
c
      return
c
   90 write(2,110) n2,maxnod
      stop
c
  110 format (1h0,10hError Stop,/,1x,16hNumber of nodes,,i5,
     1  28h, exceeds maximum dimension,,i5)
c
      end
      SUBROUTINE NODSEG (n,nseg,n1,n2,iopt)
c
c *** This subroutine finds the line segment determined by
c *** intel(n,i), (i=1,6), and divides it into nseg segments.
c *** The endpoints of these segments are stored in array node1,
c *** rows n1 through n2, and the corresponding side codes are
c *** stored in nside. If iopt = 1, the line segments are boundary
c *** nodes.  If iopt = 2, they are fracture intersections.
c
      parameter (dmin=0.0025)
      include 'maxdim.blkf/list'
      include 'comfint.blkf/list'
      include 'comnod.blkf/list'
c
      ns=nseg
      if (ns .eq. 1) go to 25
      dist=0.
      do 20 i=1,3
      dist=dist+(intel(i+3,n)-intel(i,n))**2
   20 continue
      if (dist .lt. dmin) ns=1
c
```

```
c *** Store p1 and p2
   25 n2=n1+ns-1
      do 30 i=1,3
      node1(n1,i)=inte1(i,n)
      node1(n2,i+3)=inte1(i+3,n)
   30 continue
c
      if (ns .eq. 1) go to 50
      nn=n1
      seg=real(nseg)
      do 40 i=1,nseg-1
      nn1=nn+1
      r1=real(i)
      r2=real(nseg-i)
      do 35 j=1,3
      k=j+3
      node1(nn,k)=(inte1(j,n)*r2+inte1(k,n)*r1)/seg
      node1(nn1,j)=node1(nn,k)
   35 continue
      nn=nn+1
   40 continue
c
c *** Store Side Codes
c
   50 go to (55,65),iopt
c
c *** Boundary node side codes
   55 do 60 nn=n1,n2
      nside(nn,1)=inside(1,n)
   60 nside(nn,2)=inside(2,n)
      go to 80
c
c *** Fracture intersection side codes
   65 nside(n1,1)=inside(1,n)
      nside(n2,2)=inside(2,n)
      if (ns .eq. 1) go to 80
      do 70 nn=n1,n2-1
      nside(nn,2)=0
      nside(nn+1,1)=0
   70 continue
c
   80 return
c
      end
      SUBROUTINE NORDIS (k,n1,n2,dseed,ev,sd)
c
c *** This subroutine generates random variables distributed
c *** normally with expected value ev and standard deviation 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(25/12) 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
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      real*8 dseed
      parameter (c=0.692820323)
```

```
c *** where c=1./sqrt(25./12.)
c
      do 120 i=n1,n2
      sn=0.
      do 110 j=1,25
  110 sn=sn+GGUBFS(dseed)
      sn=c*(sn-12.5)
  120 frac(k,i)=sd*sn+ev
c
      return
c
      end
      SUBROUTINE NORDIS1 (k,n1,n2,dseed,ycept,slope,sd)
c
c *** This subroutine generates random variables distributed
c *** normally with expected value ev and standard deviation sd,
c *** where ev is proportional to the logarithm of another
c *** parameter.
c
c *** Step 1.  sn = sum of 25 random variables distributed uniformly
c ***          in (0,1),
c *** Step 2.  sn = (sn-12.5)/sqrt(25/12) 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 the logarithm of another parameter,
c *** Step 4.  The value of sd*sn+ev is set so that it is never
c ***          less than a minimum value.

c
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      real*8 dseed
      parameter (c=0.692820323,emin=1.e-4)
c *** where c=1./sqrt(25./12.)
c
      do 120 i=n1,n2
      sn=0.
      do 110 j=1,25
  110 sn=sn+GGUBFS(dseed)
      sn=c*(sn-12.5)
      fl=frac(3,i)
      ev=ycept+slope*alog10(fl)
      frac(k,i)=sd*sn+ev
      if (frac(k,i).lt.emin) frac(k,i)=emin
  120 continue
c
      return
c
      end
      SUBROUTINE OVRLAP (nfrac,itole,dseed)
c
c
c     ************************************************************
c
c     This subroutine searches for fractures which lie in the
c     same plane and have centers and radii such that the
c     circles overlap.  When two such fractures are found, they
c     are combined to form one circular fracture with an area
```

```
c     equal to the combined areas and with center coordinates
c     averaged between the two original fracture centers.  The
c     apertures are either averaged or, if idist=4, NORDIS1 is
c     called to determine the aperture based on the new radius.
c
      include 'maxdim.blkf/list'
      include 'comsubi.blkf/list'
      include 'comfrc.blkf/list'
      include 'comset.blkf/list'
      include 'comref.blkf/list'
      real*8 dseed
      parameter (pi=3.141592654,cube=0.3333333333)
c
      toler = 10.**itole
c
      nfdel=0
  100 idel=1
      n1=1
      do 200 k=1,nsets
      n2=iseti(k,2)+n1-1
      if (n2.eq.nfrac) n2=nfrac-1
c
      do 190 i=n1,n2
c
c *** If fracture i was flagged for deletion go to next fracture.
      if (frac(1,i).gt.1000.) go to 190
c
c     look only at fractures in same cubes as i
c
c     loop over cubes intersected by i
c
      do 180 ientc=icubk(i),icubk(i+1)-1
      ic=icub(ientc)
c
c     loop over fractures intersecting cube ic
c
      do 180 ientf=ifcuk(ic),ifcuk(ic+1)-1
      j=ifcu(ientf)
      if(j.le.i)goto 180
c
c *** If fracture j was flagged for deletion go to next fracture.
      if (frac(1,j).gt.1000.) go to 180
c
c *** Do fractures i and j lie in the same plane?
      do 120 n=9,12
      diff = abs(frac(n,i)-frac(n,j))
      idiff = int(diff*toler)
      if (idiff.gt.0) go to 180
  120 continue
c
c *** Fractures i and j lie in the same plane.
c *** Do they overlap?
      dist=0.
      do 130 n=5,7
  130 dist=dist + (frac(n,j)-frac(n,i))**2
      dist=sqrt(dist)-(0.5/toler)
      if (dist.gt.frac(3,j)+frac(3,i)) go to 180
c
c *** Fractures i and j overlap.  Combine to form one fracture.
      frac(8,i)=frac(8,i)+frac(8,j)
```

```
      frac(3,i)=sqrt(frac(8,i)/pi)
      frac(5,i)=(frac(5,j)+frac(5,i))/2.
      frac(6,i)=(frac(6,j)+frac(6,i))/2.
      frac(7,i)=(frac(7,j)+frac(7,i))/2.
      if(iseti(k,8).eq.4)then
        ycept=rseti(k,10)
        ev=rseti(k,11)
        sd=rseti(k,12)
        call NORDIS1(4,i,i,dseed,ycept,ev,sd)
       else
        frac(4,i)=(0.5*(frac(4,j)**3
     ,                + frac(4,i)**3))**cube
       endif
c
c *** Flag fracture j for deletion.
  160 frac(1,j)=1001.
      nfdel=nfdel+1
      idel=2
c
  180 continue
c
  190 continue
      n1=n2+1
c
  200 continue
      if (idel.eq.2) go to 100
c
c *** Delete fractures which have been flagged.
c
      if (nfdel.eq.0) go to 310
      nf=0
      n1=1
      do k=1,nsets
        nfps=0
        n2=iseti(k,2)+n1-1
c
        do i=n1,n2
          ii=iref(i)
          if(frac(1,i).le.1000.)then
            nf=nf+1
            nfps=nfps+1
            iref(nf)=ii
            inew(ii)=nf
           else
            inew(ii)=0
           endif
         enddo
c
        iseti(k,2)=nfps
        n1=n2+1
c
       enddo
c
      nfrac=nf
  310 write(2,400) nfdel, nfrac
c
      return
c
c
  400 format(1h0,' the number of overlapping fractures is',i6,//,
```

```
     1       '  the number of fractures is now',i6)
c
      end
      SUBROUTINE PFS (nfrac,nflow,iray,idate,rgene,ncall)
c
c *** Calculate and print fracture statistics for each set
c
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      include 'comset.blkf/list'
      include 'comflo.blkf/list'
      include 'comref.blkf/list'
      parameter (pi=3.14159)
      dimension value(6),sumii(maxset,6),eval(maxset,6),
     '          sumij(maxset),kpar(3),npar(2,6),ndist(2,5),
     '          irkey(6),sdev(maxset,6),slope(maxset),
     '          ycept(maxset),cova(maxset),corc(maxset),
     '          itrnc(maxset)
      character idate*9,iray*7
      data kpar/2,5,6/
      data irkey/1,4,7,14,14,10/
      data npar/4hphi ,4h    ,4hthet,4ha   ,4hradi,4hus  ,4harea,
     1          4h 1  ,4harea,4h 2  ,4haper,4hture/
      data ndist/4hnorm,4hal  ,4hlogn,4horm.,4hexpo,4hn.  ,
     1           4hnorm,4hal 1,4hunif,4horm /
c
c     write titles
c
      write(2,2010)iray,idate
      if(ncall.eq.1)then
        write(2,2020)rgene
       else
         if(ncall.eq.2)then
          write(2,2030)nflow
         elseif(ncall.eq.3)then
          write(2,2040)nflow
         endif
        write(2,2050)xmesh,ymesh,zmesh
        write(2,2060)rophi,rothe
       endif
c
c     call STATOR to do simple statistics on orientation
c
      call STATOR(nfrac,eval,sdev)
c
c     initialise statistical parameters
c
      ion=0
      do iset=1,nsets
        do ipar=3,6
          eval(iset,ipar)=0.
          sdev(iset,ipar)=0.
          sumii(iset,ipar)=0.
         enddo
        if(iseti(i,8).eq.4)then
          ion=1
          sumij(iset)=0.
          corc(iset)=0.
          cova(iset)=0.
```

```
          ycept(iset)=0.
          slope(iset)=0.
         endif
        iseti(iset,2)=0
        itrnc(iset)=0
       enddo
c
      if(ncall.eq.1)then
        volume=(4./3.)*pi*rgene**3
       else
        volume=xmesh*ymesh*zmesh
       endif
c
c     do loop over the fractures
c
      do ifrac=1,nfrac
        ifold=iref(ifrac)
        iset=isetfr(ifold)
c
c       set up the value array:
c
c            1   phi
c            2   theta
c            3   radius
c            4   area (complete fractures)
c            5   area (truncations taken into account)
c            6   aperture
c
        value(1)=frac(1,ifold)
        value(2)=frac(2,ifold)
        value(3)=frac(3,ifold)
        value(4)=pi*frac(3,ifold)*frac(3,ifold)
        value(5)=frac(8,ifold)
        value(6)=frac(4,ifold)
c
        do ipar=3,6
          eval (iset,ipar)=eval (iset,ipar)+value(ipar)
          sumii(iset,ipar)=sumii(iset,ipar)+value(ipar)**2
         enddo
        if(iseti(iset,8).eq.4)
     ,    sumij(iset)=sumij(iset)+value(3)*value(6)
        iseti(iset,2)=iseti(iset,2)+1
        if(nflag(ifold).gt.0)itrnc(iset)=itrnc(iset)+1
       enddo
c
c     calculate parameters
c
      do iset=1,nsets
        if(iseti(iset,2).gt.1)then
          setsiz=float(iseti(iset,2))
          rseti(iset,13)=setsiz/volume
          do ipar=3,6
            eval(iset,ipar)=eval(iset,ipar)/setsiz
            var=sumii(iset,ipar)/setsiz-eval(iset,ipar)**2
            sdev(iset,ipar)=sqrt(abs(var))
           enddo
          if(iseti(iset,8).eq.4)then
            cova(iset)=sumij(iset)/setsiz-eval(iset,3)*eval(iset,6)
            sde36=sdev(iset,3)*sdev(iset,6)
            corc(iset)=cova(iset)/sde36
```

```
            slope(iset)=cova(iset)/sdev(iset,3)**2
            ycept(iset)=eval(iset,6)-slope(iset)*eval(iset,3)
           endif
          endif
         enddo
c
c      write results
c
c      densities
c
       if(ncall.eq.1)then
         write(2,2070)
        else
         write(2,2075)
        endif
       itrnct=0
       do i=1,nsets
         write(2,2080) i,iseti(i,2),itrnc(i),rseti(i,13)
         itrnct=itrnct+itrnc(i)
        enddo
       denst=float(nfrac)/volume
       write(2,2090)nfrac,itrnct,denst
c
c      loop over parameter types (orientation, size, and aperture)
c
       npar1=1
       do ktype=1,3
c
c        set index for iseti
c
         liset=2*ktype+1
c
c        write heading depending on ktype
c
         if(ktype.eq.1)then
           write(2,2100)
          elseif(ktype.eq.2)then
           write(2,2110)
          elseif(ktype.eq.3)then
           write(2,2120)
          endif
         if(ion.eq.0.or.ktype.ne.3)then
           write(2,2130)
          else
           write(2,2140)
          endif
c
c        loop over sets and over parameters within type
c
         do iset=1,nsets
           if(iseti(iset,2).ne.0)then
             write(2,2150)
             ichar=iseti(iset,liset)
             idist=iseti(iset,liset+1)
             do ipar=npar1,kpar(ktype)
               irs=irkey(ipar)
               if(idist.eq.4)then
                 write(2,2160)iset,iseti(iset,2),
     ,                        cova(iset),corc(iset),
     ,                        rseti(iset,irs+1),rseti(iset,irs),
```

```
     ,                    slope(iset),ycept(iset)
               else
                if(ichar.eq.1)then
                  write(2,2170)iset,iseti(iset,2),
     ,                       rseti(iset,irs+1),rseti(iset,irs+2),
     ,                       eval(iset,ipar),sdev(iset,ipar),
     ,                       npar(1,ipar),npar(2,ipar)
                 elseif(ichar.eq.2)then
                  if(ipar.eq.5)then
                    write(2,2180)iset,iseti(iset,2),
     ,                         rseti(iset,irs),
     ,                         eval(iset,ipar),sdev(iset,ipar),
     ,                         npar(1,ipar),npar(2,ipar)
                   else
                    write(2,2190)iset,iseti(iset,2),
     ,                         rseti(iset,irs),
     ,                         npar(1,ipar),npar(2,ipar)
                   endif
                 elseif(ichar.eq.3)then
                  write(2,2200)iset,iseti(iset,2),
     ,                       ndist(1,idist),ndist(2,idist),
     ,                       rseti(iset,irs+1),rseti(iset,irs+2),
     ,                       eval(iset,ipar),sdev(iset,ipar),
     ,                       npar(1,ipar),npar(2,ipar)
                 endif
               endif
             enddo
           endif
         enddo
        npar1=kpar(ktype)+1
       enddo
      return
c
 2010 format (1h1,a7,5h --- ,a9)
 2020 format('OFRACTURE STATISTICS ',
     ''AT THE END OF THE GENERATION STAGE'///
     '       ' radius of generation region:  ',f6.2/)
 2030 format('OFRACTURE STATISTICS IN FLOW REGION NUMBER ',
     'i2/' BEFORE DISCARDING ANY FRACTURE'/)
 2040 format('OFRACTURE STATISTICS IN FLOW REGION NUMBER ',
     'i2/' AFTER DISCARDING NON-CONDUCTING FRACTURES '/)
 2050 format('Osize of flow region:  ',f6.2,2('  by ',f6.2))
 2060 format(' rotation angles:      ',f6.2,' and ',f6.2/)
 2070 format('Oset      number of       number       ',
     ''density of '/
     '       '          fractures      truncated     ',
     '' fractures')
 2075 format('Oset      number of       boundary     ',
     ''density of '/
     '       '          fractures      fractures     ',
     '' fractures')
 2080 format(1h0,i2,8x,i6,9x,i6,7x,1pe10.4)
 2090 format('Ototal     ',i6,9x,i6,7x,1pe10.4)
 2100 format(//' fracture orientation angles phi and theta')
 2110 format(//' fracture radius;  fracture area   1) before ',
     '         'truncation   2) after truncation')
 2120 format(//' fracture aperture')
 2130 format('Oset  no. of  meth. of  const.  type of    ',
     ''   ev          sd            ev          sd     parameter'/
     '       '       frac.    gen.    value    dist.     ',
```

```
     ''original    original      computed    computed')
 2140 format(' -- correlation between radius and aperture',//,
     '       ' set  no. of  meth. of  covari    correl.  1/n,'
     ''        c         1/n       c'/
     2 6x,12hfrac    gen.,6x,5hance ,5x,6hcoef. ,
     3 2(3x,8horiginal),1x,2(2x,8hcomputed))
 2150 format(1h0)
 2160 format(1h ,i2,3x,i5,2x,9hgenerated,1x,e8.3e1,2x,f7.4,
     1 4(2x,e8.3e1))
 2170 format(1h ,i2,3x,i5,2x,7hread in,20x,2(1pe12.4),2x,
     1       2(1pe12.4),2x,2a4)
 2180 format(1h ,i2,3x,i5,2x,6hset to,2x,1pe12.4,30x,
     ' 2(1pe12.4),2x,2a4)
 2190 format(1h ,i2,3x,i5,2x,6hset to,2x,1pe12.4,54x,2a4)
 2200 format(1h ,i2,3x,i5,2x,9hgenerated,10x,2a4,2(1pe12.4),
     1       2x,2(1pe12.4),2x,2a4)
c
      end
      SUBROUTINE RANCEN (n1,n2,dseed,rgene,itole)
c
c *** This subroutine generates random fracture centers,
c *** (x,y,z), within a spherical generation region
c *** defined by it's radius, rgene.  The centers are
c *** stored in frac(5,i), (6,i), and (7,i).
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      real*8 dseed
c
      toler=10.**itole
      rt=2.0*rgene*toler
      rgsq=rgene*rgene
c
      do 50 i=n1,n2
   10 x=(float(int(rt*GGUBFS(dseed)))/toler) - rgene
      y=(float(int(rt*GGUBFS(dseed)))/toler) - rgene
      z=(float(int(rt*GGUBFS(dseed)))/toler) - rgene
c
      rsq = x*x + y*y + z*z
      if (rsq .gt. rgsq) go to 10
      frac(5,i)=x
      frac(6,i)=y
      frac(7,i)=z
c
   50 continue
c
      return
c
      end
      SUBROUTINE RFLOFRC (frcoef,rotf,zplane,flcoef)
c
c *** This subroutine rotates the coefficients of the
c *** flow region boundary plane equations from the
c *** global-XYZ system to a system convenient to the
c *** fracture being investigated, represented by the
c *** rotation matrix rotf, then substitutes zplane for
c *** z in the equations to get the coefficients of the
c *** equations of the lines of intersection between the
c *** boundary planes and the fracture plane.  The result
c *** is returned in array flcoef.
```

```
c
      dimension frcoef(6,4),rotf(3,3),flcoef(6,4)
c
      do 120 i=1,6
c
      do 100 j=1,3
      flcoef(i,j)=0.
c
      do 100 k=1,3
      flcoef(i,j)=flcoef(i,j)+frcoef(i,k)*rotf(k,j)
  100 continue
c
      flcoef(i,3)=flcoef(i,3)*zplane+frcoef(i,4)
      flcoef(i,4)=frcoef(i,4)
  120 continue
c
      return
c
      end
      SUBROUTINE RFRCGLO(isiz1,isiz2,xyz,k1,k2,rotf,inv)
c
c *** This subroutine transforms coordinates from the local
c *** fracture system to the global XYZ coordinate system
c *** or viceversa according to inv = 1 or 2, respectively,
c *** using the rotation matrix rotf.
c
      dimension xyz(isiz1,isiz2),rotf(3,3)
      dimension txyz(6),rot(3,3)
c
c
c *** Rotate from fracture coord. to global coord.
      if(inv.eq.1)then
        do i=1,3
          do j=1,3
            rot(i,j)=rotf(i,j)
           enddo
         enddo
       elseif(inv.eq.2)then
c
c ***    Rotate from global coord. to fracture coord.
c
         do i=1,3
           do j=1,3
             rot(i,j)=rotf(j,i)
            enddo
          enddo
        endif
c
      do k=k1,k2
        j=0
c
        do j1=1,2
          nj1=3*j1-3
          do i1=1,3
            j=j+1
            txyz(j)=0.0
            nj=nj1
            do i2=1,3
              nj=nj+1
              txyz(j)=txyz(j)+rot(i1,i2)*xyz(nj,k)
```

```
            enddo
          enddo
        enddo
c
        do n=1,6
          xyz(n,k)=txyz(n)
        enddo
c
      enddo
c
  100 return
c
      end
      SUBROUTINE ROFLOW (iside,frcoef)
c
c *** This subroutine computes the coefficients of the equations
c *** of the flow region boundary planes and stores them in the
c *** array frcoef(6,4).  Matrix rot contains the rotation matrix
c *** and the axes intercepts.
c
      parameter (pi180=0.0174532925)
      include 'comflo.blkf/list'
      dimension frcoef(6,4),iside(3,2),rot(3,4)
c
      iflag=0
      if (rophi .lt. 0.001 .and. rothe .lt. 0.001) go to 100
      iflag=1
c
      phi=rophi*pi180
      theta=rothe*pi180
c
c *** Generate rotation matrix
c
      rot(1,1)=cos(phi)*cos(theta)
      rot(1,2)=-cos(phi)*sin(theta)
      rot(1,3)=-sin(phi)
      rot(2,1)=sin(theta)
      rot(2,2)=cos(theta)
      rot(2,3)=0.
      rot(3,1)=sin(phi)*cos(theta)
      rot(3,2)=-sin(phi)*sin(theta)
      rot(3,3)=cos(phi)
  100 rot(1,4)=xmesh/2.
      rot(2,4)=ymesh/2.
      rot(3,4)=zmesh/2.
      if (iflag.eq.1) go to 120
c
c *** Generate matrix without rotation
c
      do 115 j=1,3
      do 110 k=1,3
  110 rot(j,k)=0.
      rot(j,j)=1.
  115 continue
c
c *** Fill coefficient array
c
  120 do 140 j=1,3
      l1=iside(j,1)
      l2=iside(j,2)
```

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

```
      return
c
      end
      SUBROUTINE ROJTOF (jfold,rotf,zplane,a,b,c,d,x,y,z)
c
c *** This subroutine rotates the coefficients of the equation
c *** of fracture plane jfold and the center point of fracture jfold
c *** from the global-XYZ system to a system convenient to the
c *** fracture being investigated ,nf, represented by the rotation
c *** matrix rotf. The line of intersection between planes jfold and
c *** nf is then found by substituting the plane of fracture nf
c *** (zplane) in the equation of plane jfold.
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      dimension rotf(3,3)
      dimension coefj(3),xyz(3)
c
      do 20 i=1,3
      coefj(i)=0.
      xyz(i)=0.
      do 20 j=1,3
      coefj(i)=coefj(i)+frac(j+8,jfold)*rotf(j,i)
      xyz(i)=xyz(i)+frac(j+4,jfold)*rotf(j,i)
   20 continue
c
      a=coefj(1)
      b=coefj(2)
      c=coefj(3)*zplane+frac(12,jfold)
      d=frac(12,jfold)
      x=xyz(1)
      y=xyz(2)
      z=xyz(3)
c
      return
c
      end
      SUBROUTINE SIMULC(a,b,c,xc,yc,r,nroots,x1,y1,x2,y2)
c
c *** Determine points of intersection, (x1,y1) and (x2,y2),
c *** of the given line, ax+by+c=0, and the given circle,
c *** (x-xc)**2+(y-yc)**2=r**2.  Let nroots equal the number of
c *** points of intersection, i.e. 0, 1 or 2.
c
      parameter (rmin=0.005,radmin=0.0005)
c
c *** If r equals 0 this is a point circle
      if (r .gt. rmin) go to 15
      if (abs(a*xc + b*yc + c) .lt. rmin) go to 10
      nroots=0
      go to 100
c
   10 nroots=1
      x1=xc
      y1=yc
      x2=xc
      y2=yc
      go to 100
c
   15 if ( abs(b) .lt. abs(a) ) go to 50
```

```
c
c *** Solve quadratic for x1 and x2
c
      bsq=b*b
      AQ=a*a+bsq
      BQ=2.*(a*c + a*b*yc - bsq*xc)
      CQ=bsq*(xc*xc-r*r) + (c+b*yc)**2
      rad=BQ*BQ-4.*AQ*CQ
      radnorm=rad/AQ
      if (radnorm .lt. -radmin) go to 20
      if (radnorm .gt.  radmin) go to 30
      rad=0.
      nroots=1
      go to 35
c
   20 nroots=0
      go to 100
c
   30 rad=sqrt(rad)
      nroots=2
c
   35 x1=0.5*(-BQ+rad)/AQ
      x2=0.5*(-BQ-rad)/AQ
      y1=-(a*x1+c)/b
      y2=-(a*x2+c)/b
      go to 100
c
c *** Solve quadratic for y1 and y2
c

   50 asq=a*a
      AQ=asq+b*b
      BQ=2.0*(b*c + b*a*xc - asq*yc)
      CQ=asq*(yc*yc-r*r) + (c+a*xc)**2
      rad=BQ*BQ-4.*AQ*CQ
      radnorm=rad/AQ
      if (radnorm .lt. -radmin) go to 70
      if (radnorm .gt.  radmin) go to 80
      rad=0.0
      nroots=1
      go to 85
c
   70 nroots=0
      go to 100
c
   80 rad=sqrt(rad)
      nroots=2
c
   85 y1=0.5*(-BQ+rad)/AQ
      y2=0.5*(-BQ-rad)/AQ
      x1=-(b*y1+c)/a
      x2=-(b*y2+c)/a
c
c
  100 return
c
      end
      SUBROUTINE SIMULL (a1,b1,c1,a2,b2,c2,x,y,npara)
c
c *** Solve the simultaneous equations of lines with
```

```
c *** coefficients a1,b1,c1 and a2,b2,c2.
c
      det=a1*b2-a2*b1
c
c *** If det is less than 5*10**-6, call it zero
      if ( abs(det) .lt. 0.000005 ) go to 20
c
      npara=1
      x=(b1*c2-b2*c1)/det
      y=(a2*c1-a1*c2)/det
      return
c
   20 npara=2
      return
c
      end
c
c
      subroutine STATOR(nfrac,eval,sdev)
c
c     basic statistics for orientation distributions
c
c
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      include 'comset.blkf/list'
      include 'comref.blkf/list'
c
      dimension vecg(maxset,3),xyz(maxset,3),vecm(maxset,3),
     'eval(maxset,6),sdev(maxset,6),angmax(maxset)
      pi=3.1415927
c
c     initialise with guess at mean, and zero initial vector
c
      do iset=1,nsets
        if(iseti(iset,3).eq.2)then
          phg=rseti(iset,1)
          thg=rseti(iset,4)
         else
          phg=rseti(iset,2)
          thg=rseti(iset,5)
         endif
        phg=phg*pi/180.
        thg=thg*pi/180.
        vecg(iset,1)=sin(phg)*cos(thg)
        vecg(iset,2)=-sin(phg)*sin(thg)
        vecg(iset,3)=cos(phg)
        angmax(iset)=0.
        do i=1,3
          vecm(iset,i)=0.
         enddo
       enddo
c
      do ifrac=1,nfrac
        ifold=iref(ifrac)
        iset=isetfr(ifold)
        if(iset.ne.0)then
          phi=frac(1,ifold)*pi/180.
          theta=frac(2,ifold)*pi/180.
```

```
            xyz(iset,1)=sin(phi)*cos(theta)
            xyz(iset,2)=-sin(phi)*sin(theta)
            xyz(iset,3)=cos(phi)
            si=0.
            do i=1,3
              si=si+xyz(iset,i)*vecg(iset,i)
             enddo
            is=1
            if(si.le.0.)then
              is=-1
              si=-si
             endif
            do i=1,3
              vecm(iset,i)=vecm(iset,i)+is*xyz(iset,i)
             enddo
            ang=acos(si)
            angmax(iset)=amax1(ang,angmax(iset))
           endif
         enddo
c
c     calculate mean and angular deviation
c
      do iset=1,nsets
        if(iseti(iset,2).eq.0)then
          eval(iset,1)=0.
          eval(iset,2)=0.
          sdev(iset,1)=0.
          sdev(iset,2)=0.
          goto 100
         endif
        setsiz=float(iseti(iset,2))
        vecsiz=0.
        do i=1,3
          vecsiz=vecsiz+vecm(iset,i)*vecm(iset,i)
         enddo
        vecsiz=sqrt(vecsiz)
        sdev(iset,1)=1.-(vecsiz/setsiz)
        sdev(iset,2)=sdev(iset,1)
        if(vecm(iset,3).lt.0)vecsiz=-vecsiz
        do i=1,3
          vecm(iset,i)=vecm(iset,i)/vecsiz
         enddo
        a=vecm(iset,3)
        eval(iset,1)=acos(a)
        if(vecm(iset,3).le..999)then
          a=1.-a*a
          a=vecm(iset,1)/sqrt(a)
          a=abs(a)
          eval(iset,2)=acos(a)
          if(vecm(iset,1).lt.0.)eval(iset,2)=pi-eval(iset,2)
          if(vecm(iset,2).gt.0.)eval(iset,2)=2*pi-eval(iset,2)
         else
          eval(iset,1)=0.
          eval(iset,2)=0.
         endif
c
c       check if angles are not too spread
c
        a=0.
        do i=1,3
```

```
          a=a+vecg(iset,i)*vecm(iset,i)
         enddo
        a=abs(a)
        ang=acos(a)
        ang=ang+angmax(iset)
        if(ang.ge.pi/2.)write(2,2010)iset
c
c       convert back to degrees
c
        do i=1,2
          eval(iset,i)=eval(iset,i)*180./pi
         enddo
  100   continue
       enddo
      return
 2010 format('   statistical calculations on orientation for set ',i2,
     '       ' may be off ')
      end
c
c
      SUBROUTINE STAT2D(nfrac,nflow)
c
c     calculate and print statistics on the traces
c     of the fractures on the sides of the flow
c     region.
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      include 'comset.blkf/list'
      include 'comflo.blkf/list'
      include 'comfint.blkf/list'
      include 'comref.blkf/list'
c
c     equivalence (inte1,binte),(inside,ibside),(inte2,ibinte)
c
      dimension eval(maxset,6),sumii(maxset,6),nbin(maxset,6),
     '        . ev(3),num(3),sd(3),kpar(3,3)
      data kpar/1,3,4h 1+3,2,4,4h 2+4,5,6,4h 5+6/
c
c     write heading
c
      write(2,2010)nflow
c
c     initialise statistical parameters
c
      do iside=1,6
        do iset=1,nsets
          eval(iset,iside)=0.
          sumii(iset,iside)=0.
          nbin(iset,iside)=0
         enddo
       enddo
c
c     loop over fractures
c
      do ifrac=1,nfracb
        ifold=iref(ifrac)
        int1=ibkey(ifold,2)
        int2=int1+ibkey(ifold,1)-1
        do int=int1,int2
```

```
          iset=isetfr(ifold)
          iside=-inte2(2,int)
          alen=0.
          do i=1,3
            alen=alen+(inte1(i+3,int)-inte1(i,int))**2
           enddo
          alen=sqrt(alen)
          eval(iset,iside)=eval(iset,iside)+alen
          sumii(iset,iside)=sumii(iset,iside)+alen**2
          nbin(iset,iside)=nbin(iset,iside)+1
         enddo
        enddo
c
c     calculate parameters and print results
c
      write(2,2020)
      do iset=1,nsets
        write(2,2030)
        do ior=1,3
          do k=1,3
            ev(k)=0.
            sd(k)=0.
            num(k)=0
           enddo
          do k=1,2
            iside=kpar(k,ior)
            num(k)=nbin(iset,iside)
            if(num(k).ge.1)then
              sidsiz=float(num(k))
              ev(k)=eval(iset,iside)/sidsiz
              sd(k)=sumii(iset,iside)/sidsiz
     ,                      -ev(k)**2
              sd(k)=sqrt(abs(sd(k)))
             endif
            ev(3)=ev(3)+eval(iset,iside)
            sd(3)=sd(3)+sumii(iset,iside)
            num(3)=num(3)+num(k)
           enddo
          if(num(3).ge.1)then
            sidsiz=float(num(3))
            ev(3)=ev(3)/sidsiz
            sd(3)=sd(3)/sidsiz-ev(3)**2
            sd(3)=sqrt(abs(sd(3)))
           endif
c
c         print statistics
c
          write(2,2040)iset,(kpar(k,ior),num(k),
     ,                    ev(k),sd(k),k=1,3)
         enddo
        enddo
       return
c
 2010 format(1h1//' TRACE LENGTH STATISTICS ON SIDES OF FLOW ',
     ''REGION NUMBER ',i2/' BEFORE DISCARDING ANY FRACTURE')
 2020 format(///1x,'set   ',
     '       3('side  no. of     ev          sd         '),
     '         /1x,'no.   ',
     '       3(' no.  traces   length      length       ')/)
 2030 format(1x)
```

```
 2040 format(1x,i2,4x,2(2x,i1,3x,i5,2(2x,1pe10.3)5x),
     '               a4,2x,i5,2(2x,1pe10.3))
      end
c
c
      SUBROUTINE SUBINT (nfrac,iside,itole)
c
      include 'maxdim.blkf/list'
      include 'comsubi.blkf/list'
      include 'comfrc.blkf/list'
      include 'combint.blkf/list'
      include 'comout.blkf/list'
      include 'comref.blkf/list'
      common/nodes/ifrcu(maxf2),nextc(maxf2),jfirst(maxcub),
     '             jlast(maxcub)
      dimension frcoef(6,4)
      dimension iside(3,2),rotf(3,3),flcoef(6,4)
      dimension xyz(3),ifact(3),ni(3)
c
c     determine in which subregion(s) each fracture is,
c     and set the arrays icub, icubk, ifcu, and ifcuk
c     accordingly.  Arrays ifrcu, nextc, jfirst and jlast
c     are used only temporarily.
c
      ncub=iord**3
      do i=1,ncub
        jlast(i)=0
        jfirst(i)=0
       enddo
      wids=(2.*rgene)/float(iord)
      ifact(1)=1
      ifact(2)=iord
      ifact(3)=iord**2
      ientry=0
      jj=0
c
      do i=1,6.
        do j=1,3
          frcoef(i,j)=0.
         enddo
       enddo
      do j=1,3
        l1=iside(j,1)
        l2=iside(j,2)
        frcoef(l1,j)=1.
        frcoef(l2,j)=1.
       enddo
c
c     do loop on fractures
c
      do ifrac=1,nfrac
        iref(ifrac)=ifrac
        inew(ifrac)=ifrac
        do i=1,3
          xyz(i)=frac(i+4,ifrac)+rgene
         enddo
c
c       Call subroutine ROFRAC to get the matrix of rotation, rotf,
c       the new z-plane, and the fracture center, (dk,0,zplane).
c
```

```
      call ROFRAC (ifrac,rotf,zplane,dk)
c
c
c     find the cube in which lies the center of the fracture,
c     store it, and increment key array icubk.
c
      ic=1
      do i=1,3
        a=xyz(i)/wids
        ni(i)=int(a)
        ic=ic+ni(i)*ifact(i)
       enddo
      ientry=ientry+1
      icurnt=ientry
      icubk(ifrac)=ientry
      icub(ientry)=ic
c
      do while(icurnt.le.ientry)
        ic=icub(icurnt)
        ic1=ic-1
        do i=3,1,-1
          ni(i)=ic1/ifact(i)
          ic1=ic1-ifact(i)*ni(i)
          l1=iside(i,1)
          l2=iside(i,2)
          coef=rgene-ni(i)*wids
          frcoef(l1,4)=coef
          frcoef(l2,4)=coef-wids
         enddo
c
c       find if fracture intersects boundaries of cube ic.
c
c       Call subroutine RFLOFRC to rotate the cube boundary planes
c       and determine lines of intersection with the fracture plane.
c
        call RFLOFRC (frcoef,rotf,zplane,flcoef)
c
c       set codes for intersecting fractures.
c
        rf=frac(3,ifrac)
        do i=1,3
c
          l1=iside(i,1)
          l2=iside(i,2)
c
c *** Are the boundary planes parallel to the fracture plane?
c
          a1=flcoef(l1,1)
          b1=flcoef(l1,2)
          if (iround(a1,itole).ne.0.
     ,    or.iround(b1,itole).ne.0)then
c
c *** In the plane z=zplane, does the fracture with center (dk,0)
c *** and radius rf lie between the lines l1 and l2?
c *** Compute the perpendicular distances d1 {between (dk,0) and l1},
c *** d2 {between (dk,0) and l2}, and d12 {between l1 and l2}.
c *** Compare distances rounded off to itole places.
c
            sra2b2=sqrt(flcoef(l1,1)**2+flcoef(l1,2)**2)
            d1=abs(flcoef(l1,1)*dk+flcoef(l1,3))/sra2b2
```

```
              id1=iround(d1,4)
              d2=abs(flcoef(l2,1)*dk+flcoef(l2,3))/sra2b2
              id2=iround(d2,4)
              d12=abs(flcoef(l2,3)-flcoef(l1,3))/sra2b2
              diff=d1+d2-d12
              idiff=iround(diff,4)
              irf=iround(rf,4)
c
c             if fracture crosses boundary with cube ic1,
c             check if ic1 is already in array icub, and if
c             not put it there.
c
              if(id1.lt.irf.and.ni(i).gt.0)then
                ic1=ic-ifact(i)
                do ii=icubk(ifrac),ientry
                  if(icub(ii).eq.ic1)goto 40
                 enddo
                ientry=ientry+1
                icub(ientry)=ic1
               endif
   40         if(id2.lt.irf.and.ni(i).lt.iord-1)then
                ic1=ic+ifact(i)
                do ii=icubk(ifrac),ientry
                  if(icub(ii).eq.ic1)goto 45
                 enddo
                ientry=ientry+1
                icub(ientry)=ic1
               endif
   45          continue
              endif
            enddo
           icurnt=icurnt+1
c
          enddo
c
c         increment array ifrcu for keeping track of all fractures
c         in each cube.
c
          do ii=icubk(ifrac),ientry
            jj=jj+1
            ifrcu(jj)=ifrac
            ic=icub(ii)
            jl=jlast(ic)
            if(jl.eq.0)then
              jfirst(ic)=jj
             else
              nextc(jl)=jj
             endif
            jlast(ic)=jj
           enddo
c
c         end do loop on fractures
c
         enddo
        icubk(nfrac+1)=ientry+1
c
c       set up array ifcu and key ifcuk from local array ifrcu.
c
        ientry=1
        do ic=1,ncub
```

```
          jj=jfirst(ic)
          ifcuk(ic)=ientry
          do while(jj.ne.0)
            ifcu(ientry)=ifrcu(jj)
            jj=nextc(jj)
            ientry=ientry+1
           enddo
         enddo
        ifcuk(ncub+1)=ientry+1
        ilengc=ientry
c
c     initialize negative part of array inew, used only for
c     boundary intersection side numbers.
c
        do i=-6,0
          inew(i)=i
         enddo
        return
        end
c
c
        SUBROUTINE TRUNCF (frcoef,nfrac,iside,itole)
c
        include 'maxdim.blkf/list'
        include 'comfrc.blkf/list'
        include 'combint.blkf/list'
        include 'comref.blkf/list'
        dimension frcoef(6,4)
        dimension iside(3,2),rotf(3,3),flcoef(6,4)
        dimension ntrunc(7)
        dimension bline(12,2),keybl(6,6)
        dimension axy(6,4),nxy(6,2)
c
c *** For each fracture, rotate the flow region boundary planes
c *** to a system convenient to the fracture and determine whether
c *** the fracture lies inside, partially inside, or outside the
c *** flow region.  The counter for boundary truncations is kib.
c
        kib=1
        nfracb=0
        do 90 ifr=1,nfrac
          ifrac=ifr
          ifold=iref(ifr)
c
c       Call subroutine ROFRAC to get the matrix of rotation, rotf,
c       the new z-plane, and the fracture center, (dk,0,zplane).
c
          call ROFRAC (ifold,rotf,zplane,dk)
c
c       Call subroutine RFLOFRC to rotate the flow region boundary planes
c       and determine lines of intersection with the fracture plane.n
c
          call RFLOFRC (frcoef,rotf,zplane,flcoef)
c
c
c       Determine the intersections between fracture ifrac and the
c       boundary lines, (these will be boundary nodes in the flow
c       model), and truncate the portion of the fracture that lies
c       outside the flow region boundary.  If the fracture lies
c       entirely outside the bounded region, flag the fracture for
```

```
c       discarding.
c       (ntrunc(i),i=1,6) truncation codes for the six sides,
c       ntrunc(7) = number of sides involved in truncations.
c
        rf=frac(3,ifold)
        nflag(ifold)=0
        do i=1,7
          ntrunc(i)=0
         enddo
        ibkey(ifold,1)=0
        ibkey(ifold,2)=0
c
c       Call FLAG to set codes for discarding and truncating.
c
        call FLAG(flcoef,zplane,dk,rf,iside,
     1            nflag(ifold),ntrunc,itole)
c
c
        if (nflag(ifold) .le. 0) go to 90
c
c       Call BNDINT to determine the points of intersection
c       between the six boundary lines and the end points of
c       the line segments which define the flow region in
c       the fracture plane.
c
        call BNDINT(flcoef,iside,nflag(ifold),ntrunc,
     ,              ifrac,bline,keybl)
c
c       If there is no flow region in this plane, discard
c       fracture ifrac.
c
        if (nflag(ifold) .lt. 0) go to 90
c
c       Call BNDNOD to determine all fracture/boundary intersections
c       and select those which are to be boundary nodes in the flow
c       model.  These are stored in binte with ibkey(ifrac,2) as the
c       key to the entries.
c
        call BNDNOD(ifrac,ifold,flcoef,dk,rf,zplane,ntrunc,
     1              bline,keybl,kib,axy,nxy,iside)
c
c       If there are no fracture/boundary intersections within the
c       flow region, discard the fracture.
        if (nflag(ifold).lt.0) go to 90
c
c       Call AREAT to truncate the portion of the fracture that
c       lies outside the flow region.
c
        call AREAT(dk,rf,flcoef,ntrunc,axy,nxy,flarea,ierr)
        if (ierr.ne.0)then
          write(2,110) ifrac
          stop
         endif
c
        frac(8,ifold)=flarea
   90   continue
c
      ninb=kib-1
c
  100 return
```

```
c
  110 format (1h0,' fracture',i4)
c
      end
      SUBROUTINE TRUNCG(nfrac,rgene,itole)
c
c *** This subroutine finds fractures which extend beyond the
c *** boundary of the generation region and truncates them at
c *** that boundary.  The area of the truncated region is
c *** computed and subtracted from the fracture area.  The
c *** adjusted area replaces the fracture area stored in
c *** frac(8,n).
c
c *** The plane of fracture n intersects the generation sphere
c *** in a circle, s, with center (-ad,-bd,-cd) and radius =
c *** sqrt(rgene**2-d**2), where a,b,c and d are coefficients
c *** of the equation of the plane of fracture n.
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      include 'comref.blkf/list'
      parameter (pi=3.141592654)
c
      toler=0.5/10.**itole
      do 100 n=1,nfrac
      nold=iref(n)
c
      nflag(nold)=0
      a=frac(9,nold)
      b=frac(10,nold)
      c=frac(11,nold)
      d=frac(12,nold)
      xc=frac(5,nold)
      yc=frac(6,nold)
      zc=frac(7,nold)
      rf=frac(3,nold)
      rfsq=rf*rf
c
c *** Compute the radius of circle s.
c
      rssq=rgene**2-d**2
      rs=sqrt(rssq)
c
c *** Compute the distance from the center of circle s to the
c *** center of fracture n.
c
      dfsq=(xc+a*d)**2+(yc+b*d)**2+(zc+c*d)**2
      df=sqrt(dfsq)
c
c *** Does fracture n extend beyond the boundary?
c *** If not, go on to the next fracture.
c
      if ((rf+df).le.(rs+toler)) go to 100
      if ((rf-df).ge.(rs-toler)) go to 90
c
c *** Compute the area of truncation.
c
      beta=acos((dfsq+rfsq-rssq)/(2.*df*rf))
      alpha=2.*acos((dfsq+rssq-rfsq)/(2.*df*rs))
      gamma=2.*(pi-beta)
```

```
      atrunc=0.5*rfsq*(gamma-sin(gamma))-0.5*rssq*(alpha-sin(alpha))
c
c *** Compute the truncated area of fracture n.
c
      frac(8,nold)=frac(8,nold)-atrunc
      go to 95
   90 frac(8,nold)=pi*rssq
c
c *** Flag the truncated fracture.
c
   95 nflag(nold)=1
c
  100 continue
c
      return
c
      end
      SUBROUTINE TRUNCN (nfold,flcoef,iside,nbn,njold,aj,bj,cj,
     1                   xy,nside,node)
c
c *** Given fracture/fracture node, nfold/njold, with endpoints
c *** (xy(3,1),xy(3,2)) and (xy(4,2),xy(4,2)) on line lj and
c *** nbn boundary nodes on fracture nfold, this subroutine
c *** truncates node nfold/njold at any or all of the nbn nodes.
c *** The variable 'node' is returned as a flag :
c *** node=0  no node, node=1 untruncated node,
c *** node=2  truncated node.
c *** track of the truncated part of the node is kept in array
c *** endout.
c
      include 'maxdim.blkf/list'
      include 'combint.blkf/list'
      include 'comout.blkf/list'
      dimension flcoef(6,4),iside(3,2),xy(4,2)
      dimension xyn(3,2),nside(2)
c
      node=1  .
      nside(1)=0
      nside(2)=0
      nnt=0
      ibk=ibkey(nfold,2)-1
c
      do 90 nb=1,nbn
      ibk=ibk+1
      l1=ibside(1,ibk)
      l2=0
c
c *** Find parallel boundary line to boundary node
      do 20 k=1,3
      if (l1 .eq. iside(k,1)) l2=iside(k,2)
      if (l1 .eq. iside(k,2)) l2=iside(k,1)
      if (l2 .gt. 0) go to 22
   20 continue
c
c *** Determine intersection of line lj with boundary
c *** lines l1 and l2.
   22 lk=l1
      do 28 k=1,2
      a=flcoef(lk,1)
      b=flcoef(lk,2)
```

```
      c=flcoef(lk,3)
      call SIMULL(aj,bj,cj,a,b,c,x,y,npara)
      go to (24,70),npara
   24 xy(k,1)=x
      xy(k,2)=y
      lk=l2
   28 continue
c
c
c *** Line lj is not parallel to boundary lines l1 and l2.
c *** Determine the relative positions of the four points
c *** stored in array xy.
   30 call COMPT(xy,4,1,2,3,loc3)
      call COMPT(xy,4,1,2,4,loc4)
      loc4p2=loc4+2
      go to (31,33,33,35,35),loc3+2
   31 go to (99,42,42,90,90),loc4p2
   33 go to (40,99,99,40,40),loc4p2
   35 go to (90,42,42,90,90),loc4p2
c
   40 in=3
      go to 45
   42 in=4
   45 nout=nout+1
      endout(1,nout)=xy(in,1)
      endout(2,nout)=xy(in,2)
      endout(4,nout)=xy(1,1)
      endout(5,nout)=xy(1,2)
      ifout(nout,1)=nfold
      ifout(nout,2)=njold
      xyn(3,1)=xy(1,1)
      xyn(3,2)=xy(1,2)
      xy(in,1)=xy(1,1)
      xy(in,2)=xy(1,2)
c
c *** Truncate fracture node at boundary node?
      xyn(1,1)=binte(1,ibk)
      xyn(1,2)=binte(2,ibk)
      xyn(2,1)=binte(4,ibk)
      xyn(2,2)=binte(5,ibk)
      call COMPT(xyn,3,1,2,3,loc)
      if (loc .le. 0) go to 90
c
c *** Truncate fracture node
      nside(in-2)=l1
      nnt=nnt+1
      if (nnt .gt. 2) go to 125
      node=2
      go to 90
c
c *** Line lj is parallel to boundary lines l1 and l2.
c *** Determine intersections between lines l1 and l2 and
c *** a perpendicular line through (xy(3,1),xy(3,2)), a
c *** point on line lj, in order to tell whether line lj
c *** is inside boundary line l1.
c
   70 aperp=-bj
      bperp= aj
      cperp=bj*xy(3,1)-aj*xy(3,2)
c
```

```
      do 72 k=1,2
      call SIMULL (a,b,c,aperp,bperp,cperp,x,y,npara)
      if (npara .eq. 2) go to 120
      xy(k,1)=x
      xy(k,2)=y
      c=flcoef(l2,3)
   72 continue
c
      call COMPT (xy,4,1,2,3,loc)
      go to (99,99,99,90,90),loc+2
c
   90 continue
      return
c
   99 node=0
c     increment array for keeping track of intersections outside
c     of the flow region (used for plotting program input).
c
      nout=nout+1
      endout(1,nout)=xy(3,1)
      endout(2,nout)=xy(3,2)
      endout(4,nout)=xy(4,1)
      endout(5,nout)=xy(4,2)
      ifout(nout,1)=nfold
      ifout(nout,2)=njold
  100 return
c
  120 write(2,130) nfold,njold
      stop
c
  125 write(2,135) nfold,njold
      stop
c
  130 format (1h0,'Error, line lj is parallel to its perpendicular',/,
     1         1x,'subroutine truncn, fractures',i5,' and',i5)
  135 format (1h0,'Error, intersection between fractures',i5,' and',
     1    i5,/,1x,'is truncated at more than two boundary nodes.')
c
      end
      SUBROUTINE UNIDIS (k,n1,n2,dseed,center,range)
c
c *** This subroutine generates random variables distributed
c *** uniformly over [(center-range) , (center+range)].
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      real*8 dseed
      parameter (pi=3.1415927)
c
c
c     if parameter is phi, add correction factor to get
c     a uniform distribution on the sphere.
c
      if(k.eq.1)then
        phi1=(center-range)*pi/180.
        phi2=(center+range)*pi/180.
        amin=1-cos(phi1)
        amax=1-cos(phi2)
        if(phi1.lt.0.)amin=-amin
        center=(amin+amax)/2.
```

```
        range=(amax-amin)/2.
       endif
c
      do i=n1,n2
        frac(k,i)=center+(2*GGUBFS(dseed)-1)*range
c
c        correction for phi
c
         if(k.eq.1)then
           if(frac(k,i).ge.0.)then
             frac(k,i)=acos(1.-frac(k,i))*180./pi
            else
             frac(k,i)=-acos(1.+frac(k,i))*180./pi
            endif
          endif
c
       enddo
      return
c
      end
c
      SUBROUTINE WDIMES (nfrac,title,nflow)
c
c *** This subroutine writes file 8 for input to DIMES.
c
      include 'maxdim.blkf/list'
      include 'comfint.blkf/list'
      include 'comfrc.blkf/list'
      include 'comflo.blkf/list'
      include 'comnod.blkf/list'
      include 'comout.blkf/list'
      include 'comref.blkf/list'
      include 'comsubi.blkf/list'
      dimension numinc(maxcub),numfrc(maxcub)
      character file*11
      integer title(40)
c
      write(file,40)nflow
   40 format('dimes',i2.2,'.dat')
      open(unit=8,file=file,status='new',shared)
      write(8,8000) (title(i),i=1,40)
      xgene=2*rgene
      ygene=2*rgene
      zgene=2*rgene
      write(8,8010) xgene,ygene,zgene
      write(8,8010) xmesh,ymesh,zmesh
      write(8,8010) rophi,rothe
      nbpt=inst+nint
      write(8,8020) nfrac,nbpt,nout
      write(8,8020) iord
      radmax=0.
      apmax=0.
      do ic=1,ncub
        numint=0
        numf=0
        do ientf=ifcuk(ic),ifcuk(ic+1)-1
          ifold=ifcu(ientf)
          if(inew(ifold).ne.0)then
            numf=numf+1
            numint=numint+ibkey(ifold,1)
```

```
            ni=inkey(ifold,1)
            if (ni .gt. 0)then
              ks=inkey(ifold,2)-1
              do 30 n=1,ni
   20           ks=ks+1
                k=inte3(ks)
                if(k.eq.0)goto 20
                k=k+inst
                in1=inte2(1,k)
                in2=inte2(2,k)
                if(in1.ne.ifold)then
                  do icc=icubk(in1),icubk(in1+1)-1
                    if(icub(icc).eq.ic)goto 30
                  enddo
                 endif
                numint=numint+1
   30          continue
             endif
            radmax=max(radmax,frac(3,ifold))
            apmax=max(apmax,frac(4,ifold))
           endif
         enddo
        numinc(ic)=numint
        numfrc(ic)=numf
       enddo
c
      write(8,8030)apmax,radmax
c
      write(8,8040)(numfrc(ic),ic=1,ncub)
      write(8,8040)(numinc(ic),ic=1,ncub)
      write(8,8040)( ifrok(ic),ic=1,ncub+1)
      do ic=1,ncub
        write(8,8040)ic
        do ientf=ifcuk(ic),ifcuk(ic+1)-1
          ifold=ifcu(ientf)
          ifrac=inew(ifold)
          if(ifrac.ne.0)write(8,8050)
     ,                    ifrac,(frac(j,ifold),j=1,7)
         enddo
        if(ifrok(ic).ne.ifrok(ic+1))then
          do ient=ifrok(ic),ifrok(ic+1)-1
            iout=ifro(ient)
            if1=ifout(iout,1)
            if2=ifout(iout,2)
            write(8,8060)iout,(endout(j,iout),j=1,6),
     ,                    inew(if1),inew(if2)
           enddo
         endif
        do ientf=ifcuk(ic),ifcuk(ic+1)-1
          ifold=ifcu(ientf)
          ifrac=inew(ifold)
          if(ifrac.ne.0)then
            if (ibkey(ifold,1).ne.0)then
              k=ibkey(ifold,2)
              do n=1,ibkey(ifold,1)
                in1=inte2(1,k)
                in2=inte2(2,k)
                write(8,8060)k,(intel(j2,k),j2=1,6),
     ,                       inew(in1),inew(in2)
                k=k+1
```

```
              enddo
             endif
c
            ni=inkey(ifold,1)
            if (ni .gt. 0)then
              ks=inkey(ifold,2)-1
              do 60 n=1,ni
   50           ks=ks+1
                k=inte3(ks)
                if(k.eq.0)goto 50
                k=k+inst
                in1=inte2(1,k)
                in2=inte2(2,k)
                if(in1.ne.ifold)then
                  do icc=icubk(in1),icubk(in1+1)-1
                    if(icub(icc).eq.ic)goto 60
                  enddo
                 endif
                write(8,8060)k,(inte1(j2,k),j2=1,6),
     ,                          inew(in1),inew(in2)
  60           continue
             endif
           endif
         enddo
       enddo
c
      close(unit=8)
      return
c
 8000 format (20a4)
 8010 format (4f10.4)
 8020 format (3i10)
 8030 format (2(1pe10.3))
 8040 format (10i6)
 8050 format(i6,7(1pe10.3))
 8060 format (i6,6f10.4,2i10)
c
      end
c
      SUBROUTINE WFOUR(nfrac)
c
c *** Write file 4 to checkout subroutine ovrlap.
c
c     SUBROUTINE NOT CALLED ANY MORE
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      include 'comset.blkf/list'
      include 'comref.blkf/list'
c
c
      write(4,100)
      write(4,110)
      write(4,120)
      i=1
      do 30 n=1,nsets
      m=iseti(n,2)
      write(4,130) n,m
      if (m.eq.0) go to 30
      do 20 l=1,m
```

```
      write(4,140) i,(frac(j,iref(i)),j=1,8)
      i=i+1
   20 continue
   30 continue
c
      write(4,150)
      do 40 n=1,nfrac
      write(4,160) n,(frac(j,iref(n)),j=9,12)
   40 continue
c
      return
c
c
  100 format(1h1,' ***** Printout to check subroutine ovrlap *****',//)
  110 format(1h0,25h Fracture Characteristics,/,
     1       37h id.  orientation    radius  aperture,
     2       35h   coordinates of center   fracture)
  120 format(1h ,18h no.   phi   theta,24x,1hx,8x,1hy,8x,1hz,5x,4harea)
  130 format(1h0,' set ',i3,' contains ',i3,' fractures')
  140 format(1h ,i4,2f7.2,f9.3,e10.2,3f9.3,e10.2)
  150 format(1h0,' Coefficients of fracture plane equations',//)
  160 format(1h ,i10,4f12.4)
c
      end
      SUBROUTINE WFRAC1 (iopt,nfrac,rgene,iray,idate)
c
c *** This subroutine writes fracture characteristics stored
c *** in array frac by sets on unit 4 depending on
c *** option chosen (write if idump=1).
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      include 'comset.blkf/list'
      include 'comflo.blkf/list'
      include 'comref.blkf/list'
      character iray*7,idate*9
c
c
      write(4,150) iray,idate
      go to (10,20,30,30),iopt
c
   10 write(4,180)
      write(4,200) nfrac
      write(4,160) nsets
      do 15 i=1,nsets
      write(4,170) i,iseti(i,2),rseti(i,13)
   15 continue
      go to 60
c
   20 write(4,210)
      write(4,190) rgene
      write(4,205) nfrac
      go to 60
c
   30 write(4,240)
      if (iopt.eq.3) write(4,250)
      if (iopt.eq.4) write(4,270)
      write(4,260) xmesh,ymesh,zmesh,rophi,rothe
      write(4,280) nfrac
c
```

```
   60 write(4,110)
      write(4,120)
      i=1
      do 80 ifrac=1,nfrac
      ifold=iref(ifrac)
      write(4,140) ifrac,(frac(j,ifold),j=1,8),isetfr(ifold)
   80 continue
c
      if (iopt.ne.2) go to 100
      write(4,220)
      do 90 i=1,nfrac
      iold=iref(i)
      if (nflag(iold).eq.0) go to 90
      write(4,230) i
   90 continue
c
c
  100 return
c
c
  110 format(1h0,25h Fracture Characteristics,/,1x,
     1       37h id.   orientation   radius  aperture,
     2       43h   coordinates o  f center   fracture   set)
  120 format(1h ,18h no.   phi   theta,25x,1hx,8x,1hy,8x,1hz,7x,4harea)
  130 format(1x)
  140 format(1h ,i4,2f7.2,f9.3,e10.2,3f9.3,e11.3,i4)
  150 format(1h1,a7,5h --- ,a9)
  160 format(1h0,52hThe number of fracture sets generated or read in is
     1      ,i3,/////,6x,3hset,6x,19hnumber of fractures,6x,
     2       20hdensity of fractures,/)
  170 format(1h0,2x,i5,13x,i5,19x,e10.4)
  180 format(1h0,37hf r a c t u r e   g e n e r a t i o n)
  190 format(1h0,39hthe radius of the generation region is ,f8.4)
  200 format(1h0,48hthe number of fractures generated or read in is ,i4)
  205 format(1h0,51hthe number of fractures in the generation region is,
     1        i4)
  210 format(1h0,78ht r u n c a t e d   f r a c t u r e s   o f   g e n
     1e r a t i o n   s t a g e )
  220 format(1h0,49hFractures truncated at generation region boundary,/)
  230 format(1h ,10x,i5)
  240 format(1h0,47hf r a c t u r e s   i n   f l o w   r e g i o n)
  250 format(1h ,47h( a l l   f r a c t u r e s   i n c l u d e d ))
  260 format(1h0,31hthe size of the flow region is ,f8.4,4h by ,f8.4,
     1  4h by,f8.4,/,1x,27hthe angles of rotation are ,f6.2,5h and ,
     2  f6.2)
  270 format(1h0,50h( n o n - c o n d u c t i n g   f r a c t u r e s ,
     1          43h  h a v e   b e e n   e l i m i n a t e d ))
  280 format(1h0,46hthe number of fractures in the flow region is ,i4)
c
      end
      SUBROUTINE WINTER (nfrac,iopt)
c
c *** This subroutine writes the boundary nodes stored in binte
c *** and intersections stored in intel, (if iopt=2) on unit 4.
c
c *** If iopt=4, fracture network information is written on
c *** unit 7 for Program IPLANE.
c
c
      include 'maxdim.blkf/list'
```

```
      include 'comfint.blkf/list'
      include 'comref.blkf/list'
c     equivalence (intel,binte),(inside,ibside),(inte2,ibinte)
c
      if(iopt.eq.4)goto 75
      write(4,110)
      if (iopt .eq. 1) write(4,115)
      if (iopt .eq. 2) write(4,120)
      if (iopt .gt. 2) write(4,500)
c *** Write boundary nodes.  Note: intel=binte,
c *** inte2=ibinte, inside=ibside.
      do i=1,nfrac
        iold=iref(i)
        if (ibkey(iold,1).ne.0)then
          write(4,130)
          k=ibkey(iold,2)
          do n=1,ibkey(iold,1)
            in1=inte2(1,k)
            in2=inte2(2,k)
            write(4,140) i,(inside(j1,k),j1=1,2),
     1                     (intel(j2,k),j2=1,6),
     2                     inew(in1),inew(in2)
            k=k+1
          enddo
        endif
      enddo
      go to (80,50,50),iopt
c
   50 write(4,150)
      write(4,120)
      do i=1,nfrac
        iold=iref(i)
        ni=inkey(iold,1)
        if(ni.gt.0)then
          write(4,130)
          ks=inkey(iold,2)-1
          do n=1,ni
   55       ks=ks+1
            k=inte3(ks)
            if(k.eq.0)goto 55
            k=k+inst
            in1=inte2(1,k)
            in2=inte2(2,k)
            write(4,140) i,(inside(j1,k),j1=1,2),
     1                     (intel(j2,k),j2=1,6),
     2                     inew(in1),inew(in2)
          enddo
        endif
      enddo
c
      return
c
c *** Write intersection data for program PLANE
c
   75 nbpt=inst+nint
      write(7,700) nfrac,ninb,nint
      write(7,705)
      write(7,710) ((ibkey(iref(i),j),j=1,2),i=1,nfrac)
      write(7,711)
      write(7,710) ((inkey(iref(i),j),j=1,2),i=1,nfrac)
```

```
      write(7,712)
      write(7,710) ((inside(j,i),j=1,2),i=1,nbpt)
      write(7,713)
      write(7,710) ((inew(inte2(j,i)),j=1,2),i=1,nbpt)
      write(7,714) inte3(1)
      write(7,730) (inte3(i),i=2,inte3(1))
      write(7,715)
      write(7,720) ((inte1(j,i),j=1,6),i=1,nbpt)
c
   80 return
c
  110 format (1h1,44h Fractures truncated at flow region boundary)
  115 format (1h0,34hfrac side       endpoints of node ,
     1 14h(local coord.),20x,9hfractures,
     2 /,2x,9hno. codes,6x,2hx1,8x,2hy1,
     3 8x,2hz1,8x,2hx2,8x,2hy2,8x,2hz2,3x,7hnf1 nf2)
c 120 format (1h0,34hfrac side       endpoints of node ,
c    1 15h(global coord.), /,2x,10hno. codes ,5x,2hx1,8x,2hy1,
c    2 8x,2hz1,8x,2hx2,8x,2hy2,8x,2hz2)
  120 format (1h0,34hfrac side       endpoints of node ,
     1 15h(global coord.),19x,9hfractures,
     2  /,2x,10hno. codes ,5x,2hx1,8x,2hy1,
     3 8x,2hz1,8x,2hx2,8x,2hy2,8x,2hz2,3x,7hnf1 nf2)
  130 format (1x)
  140 format (1h ,i4,2i3,6f10.3,2i4)
  150 format (1h0,31hFracture/fracture intersections,//)
  500 format (1h0,'****** for checkout of intfra and discar ******')
  700 format (7h nfrac ,i5,10h     ninb ,i5,10h     nint ,i5)
  705 format (7h ibkey )
  710 format (5(2i5,5x))
  711 format (7h inkey )
  712 format (7h inside)
  713 format (7h inte2 )
  714 format (9h inte3   ,i5,9h entries )
  715 format (7h inte1 )
  720 format (6f10.4)
  730 format (15i5)
      end
      SUBROUTINE WNODES (frcoef,nfrac,nseg,nsegb,iray,idate,title)
c
c *** This subroutine writes file 6 for input to DISCEL
c
      include 'maxdim.blkf/list'
      include 'comfint.blkf/list'
      include 'comfrc.blkf/list'
      include 'comflo.blkf/list'
      include 'comnod.blkf/list'
      include 'comref.blkf/list'
      dimension frcoef(6,4)
      character iray*7,idate*9
      integer title(40)
c
      write(6,6000) iray,idate
      write(6,6010) (title(i),i=1,40)
      write(6,6020) xmesh,ymesh,zmesh
      write(6,6030) rophi,rothe
      write(6,6040)
      write(6,6050) ((frcoef(i,j),j=1,4),i=1,6)
      write(6,6060) nfrac
      write(6,6070)
```

```
      do i=1,nfrac
        iold=iref(i)
        write(6,6080) (frac(j,iold),j=1,12)
       enddo
      write(6,6090) nnode
      write(6,6100) nseg,nsegb
      write(6,6110)
      write(6,6120) ((node1(i,j),j=1,6),i=1,nnode)
      write(6,6130)
      write(6,6140) ((nside(i,j),j=1,2),i=1,nnode)
      write(6,6150)
      write(6,6140) ((node2(i,j),j=1,2),i=1,nnode)
      k=node3(1)
      write(6,6160) k
      write(6,6170) (node3(i),i=2,k)
      write(6,6180)
      write(6,6140) ((nkey(i,j),j=1,2),i=1,nfrac)
      return
c
c
 6000 format (a7,5h --- ,a9)
 6010 format (20a4)
 6020 format (10hxmesh- - -,f10.4,10hymesh- - -,f10.4,
     1        10hzmesh- - -,f10.4)
 6030 format (10hrophi- - -,f10.4,10hrothe- - -,f10.4)
 6040 format (10h    frcoef)
 6050 format (4f10.4)
 6060 format (10hnfrac- - -,i5)
 6070 format (10h    frac  )
 6080 format (7f10.4,e12.4,4e12.4)
 6090 format (10hnnode- - -,i5)
 6100 format (10h    nseg  ,i5,10h    nsegb ,i5)
 6110 format (10h    node1 )
 6120 format (6f10.4)
 6130 format (10h nside    )
 6140 format (5(3x,2i6))
 6150 format (10h node2    )
 6160 format (9h node3   ,i6,8h entries)
 6170 format (15i5)
 6180 format (10h nkey     )
      end
c
c
c
c     SUBROUTINE WPLOT (iopt,nfrac)
c
c     This subroutine rotates boundary nodes and fracture
c     intersections, iopt=1, or all nodes, iopt=2, from
c     global to local coordinates and writes the endpoints
c     on unit 3 for input to a plotting program.
c     The maximum radius is computed for scaling plots.
c
c     SUBROUTINE NOT CALLED ANY MORE
c
c     include 'maxdim.blkf/list'
c     include 'comfrc.blkf/list'
c     include 'comfint.blkf/list'
c     include 'comnod.blkf/list'
c     include 'comref.blkf/list'
c     dimension xy(6,maxnpf),kxy2(maxnpf),kside(maxnpf,2)
```

```
c     dimension rotf(3,3)
c
c
c     nplot=iopt
c     write(3,160) nplot
c     write(3,150) nfrac
c
c     rmax=0.0
c     do 5 nf=1,nfrac
c     nfold=iref(nf)
c     rmax=max(rmax,frac(3,nfold))
c   5 continue
c     write(3,170) rmax
c
c     do 90 nff=1,nfrac
c     nf=nff
c     nfold=iref(nf)
c     rf=frac(3,nfold)
c     Call ROFRAC for rotation matrix, rotf.
c     call ROFRAC(nfold,rotf,zplane,dk)
c
c     k=0
c     go to (10,50),iopt
c
c *** Accumulate boundary nodes on fracture nf.
c  10 nbn=ibkey(nfold,1)
c     if (nbn .eq. 0) go to 35
c     n1=ibkey(nfold,2)
c     n2=n1+nbn-1
c
c     do 30 n=n1,n2
c     k=k+1
c     if (k .gt. maxnpf) go to 95
c     do 20 j=1,6
c  20 xy(j,k)=inte1(j,n)
c
c     njold=inte2(1,n)
c     if (njold .eq. nfold) njold=inte2(2,n)
c     kxy2(k)=inew(njold)
c
c     kside(k,1)=inside(1,n)
c     kside(k,2)=inside(2,n)
c  30 continue
c
c *** Accumulate fracture intersections, iopt=1.
c  35 nn=inkey(nfold,1)
c     if (nn .eq. 0) go to 80
c     n1=inkey(nfold,2)
c     n2=n1+nn-1
c
c     do 45 n=n1,n2
c     k=k+1
c     if (k .gt. maxnpf) go to 95
c     nk=inte3(n)+ninb
c     do 40 j=1,6
c  40 xy(j,k)=inte1(j,nk)
c
c     njold=inte2(1,nk)
c     if (njold .eq. nfold) njold=inte2(2,nk)
c     kxy2(k)=inew(njold)
```

```
c
c     kside(k,1)=inside(1,nk)
c     kside(k,2)=inside(2,nk)
c  45 continue
c     go to 80
c
c  50 nn=nkey(nfold,1)
c     if (nn .eq. 0) go to 80
c     n1=nkey(nf,2)
c     n2=n1+nn-1
c
c     do 70 n=n1,n2
c     k=k+1
c     if (k .gt. maxnpf) go to 97
c     nk=node3(n)
c     do 60 j=1,6
c  60 xy(j,k)=node1(nk,j)
c
c     njold=node2(nk,1)
c     if (njold .eq. nfold) njold=node2(nk,2)
c     kxy2(k)=inew(njold)
c
c     kside(k,1)=nside(nk,1)
c     kside(k,2)=nside(nk,2)
c  70 continue
c
c  80 write(3,110) nf,rf,dk,k
c     if (k .eq. 0) go to 90
c
c *** Rotate boundary nodes and intersections or nodes from
c *** global to local coordinates for plotting.
c     call RFRCGLO (6,maxnpf,xy,1,k,rotf,2)
c     write(3,120) ((xy(j,i),j=1,6),i=1,k)
c     write(3,130) (kxy2(i),i=1,k)
c     write(3,140) ((kside(i,j),j=1,2),i=1,k)
c
c  90 continue
c     return
c
c  95 write(2,180) nf,k,maxnpf
c     stop
c
c  97 write(2,190) nf,k,maxnpf
c     stop
c
c 100 return
c
c
c 110 format (i5,2f10.4,i5)
c 120 format (6f10.4)
c 130 format (10i5)
c 140 format (5(2i5,5x))
c 150 format (10hnfrac --- ,i5)
c 160 format (10hnplot --- ,i5)
c 170 format (10hmax radius,f10.4)
c 180 format (1h0,'***** ERROR STOP SUBROUTINE WPLOT *****',/,
c    1 ' fracture ',i5,' contains ',i5,' intersections',/,
c    2 ' maximum per fracture =',i5)
c 190 format (1h0,'***** ERROR STOP SUBROUTINE WPLOT *****',/,
c    1 ' fracture ',i5,' contains ',i5,' nodes',/,
```

```
c    2 ' maximum per fracture =',i5)
c
c     end
      SUBROUTINE WRFIVE (iopt,iray,idate,title,itole,nfrac,rgene)
c
c *** This subroutine either writes or reads all primary fracture
c *** system data onto or from unit 5.
c ***    if iopt = 1 then write
c ***    if iopt = 2 then read
c
      include 'maxdim.blkf/list'
      include 'comfrc.blkf/list'
      include 'comset.blkf/list'
      include 'comref.blkf/list'
      include 'comsubi.blkf/list'
      integer title(40)
      character iray*7,idate*9
c
c
      if(iopt.eq.1)then
        open(unit=5,file='frac3d.dat',status='new',form=
     '  'unformatted',shared)
        write(5) iray,idate
        write(5) (title(i),i=1,40)
        write(5) nsets,itole
        write(5) ((iseti(i,j),j=1,8),i=1,nsets)
        write(5) ((rseti(i,j),j=1,16),i=1,nsets)
        write(5) nfrac,rgene
        do i=1,nfrac
          iold=iref(i)
          write(5) iold,(frac(j,iold),j=1,12),
     '               isetfr(iold),nflag(iold),
     '               icubk(iold)
         enddo
        write(5) ncub,ilengc
        write(5) (ifcuk(i),i=1,ncub)
        write(5) (icub(i),ifcu(i),i=1,ilengc)
       else
        open(unit=5,file='frac3d.dat',status='old',form=
     '  'unformatted',shared)
        read (5) iray,idate
        read (5) (title(i),i=1,40)
        read (5) nsets,itole
        read (5) ((iseti(i,j),j=1,8),i=1,nsets)
        read (5) ((rseti(i,j),j=1,16),i=1,nsets)
        read (5) nfrac,rgene
        do i=1,nfrac
          read (5) iold,(frac(j,iold),j=1,12),
     '               isetfr(iold),nflag(iold),
     '               icubk(iold)
          iref(i)=iold
          inew(iold)=i
         enddo
        read (5) ncub,ilengc
        read (5) (ifcuk(i),i=1,ncub)
        read (5) (icub(i),ifcu(i),i=1,ilengc)
       endif
      close(unit=5)
      return
      end
```

APPENDIX C

Program DISCEL Organization and Arrays

Table C-1. Description of DISCEL Program Variables

Variable	Description	How Value is Assigned
Global:		
i	Local node number of current node on current fracture,nf.	Set to do-loop index inode
ier	Error flag	Set by various subroutines
inew	Flag for reading boundary conditions: = 1 before EOF encountered, continue to read data Groups 2 and 3; = 2 after EOF encountered, discontinue reading data Groups 2 and 3, use last boundary conditions for all subsequent fracture meshes.	set in DISCEL
inode	Nodal do-loop index, inode = 1,nt.	Set in DISCEL
istore	Flag for storing G matrix of current fracture,nf = 0 do not store G matrix; = 1 store G matrix, fracture contains one or more boundary nodes.	Set in ROFRAC
j	Local node number of node on fracture nf being integrated w.r.t. node i.	Set in do-loop in INTEGFN,INTEGFP, INTEGFR,FORMG, SFLUX,STOREG
kstore	Counter for linear array Gstore, (locally,k)	Computed in STOREG
maxd	The number of flow directions input, equals 1, 2, or 3.	Set in BNDCON
maxfrc	Dimensioning constant, the maximum number of fractures.	Set in included parameter statement
maxh	Dimensioning constant, the maximum number of constant head values assigned, must be at least maxd times the number of boundary nodes.	Set in included parameter statement
maxn	Dimensioning constant, the maximum number of nodes per fracture, can not be greater than(maxnod - 9).	Set in included parameter statement
maxnd	Duplicate variable for constant maxn.	Set in DISCEL
maxnod	Dimensioning constant, the maximum number of nodes.	Set in included parameter statement

mdir	Duplicate variable for maxd, which is in a common block.	Set in DISCEL
mfrac	Duplicate variable for constant maxfrc.	Set in DISCEL
mnode	Duplicate variable for nnode, which is in a common block.	Set in DISCEL
mnod3	Dimensioning constant equal to maxnod * 3.	Set in parameter statement in DISCEL
nbcs	Counter for print out of the number of times boundary conditions, Group 3, have been read for current mesh.	Computed in DISCEL
ndata	Flag set when reading fracture mesh data file from FMG3D: = 1 complete data set read; = 2 normal end of data and EOF; = 3 incomplete data set and EOF.	Set in READ2
ndate	Program run date, character variable *9	Set in call to VAX/ VMS subroutine DATE
new	Flag for reading next boundary conditions: > 0 read Group 3 = 0 use previous Group 3.	Set in DISCEL modified in BNDCON
nf	Number of current fracture, set to do-loop index nfr.	Set in DISCEL
nfr	Main fracture do-loop index, nfr = 1,nfrac.	Set in DISCEL
nmesh	Counter for print out of number of the current fracture mesh.	Computed in DISCEL
nnfrac	Duplicate variable for nfrac, for storage in a separate common block.	Set in READ2
nnpf	Duplicate variable for nt, which is in a common block.	Set in DISCEL
nonz	Maximum number of non-zero elements in global mass-balance matrixA	Set in parameter statement in DISCEL
nray	The program name, DISCEL, for print out, character variable *7	Set in DISCEL data statement
nt	The number of nodes on the current fractue, nf; nt = nkey(nf,1).	Set in ROFRAC, BWIDTH, SFLUX
n1	The starting location in array node3 of the global node numbers of the nt nodes on fracture nf; n1 = nkey(nf,2).	Set in ROFRAC, BWIDTH, SFLUX
n2	The ending location in array node3 of the global node numbers of the nt	Computed in ROFRAC, BWIDTH, SFLUX

	nodes on fracture nf; n2 = n1 + nt - 1.	
pkb	Transmissivity of the current fracture, nf; pkb = rogomu * (aperture)** 3.	Computed in ROFRAC
radius	The radius of the current fracture, nf; radius = frac1(nf,3).	Set in ROFRAC
rogomu	Permeability fracture $\rho g/12\mu$ in cgs of mks units.	Set in DISCEL
Local		
calphx cbetax	Rotation parameters cos(alphax) and cos(betax) from the xy-coordinate system of the current fracture, nf, to the x′ y′ -coordinate system of a radial or nonradial node.	Computed in RADIAL or NRADIAL
p	The perpendicular distance from a nonradial node to the fracture center, i.e., the x′ -coordinate of the node.	Computed in NRADIAL
phi	An orientation angle of the current fracture, nf, in radians; phi = frac1(nf,1) * (pi/180).	Computed in ROFRAC
pi	The constant pi = 3.141592654.	Used in parameter statements
SF0	The shape function for the sink image of node i w.r.t. nodes j=1,nt $SF0 = \sum_{j=1}^{nt} f_{iI-}^{k}$	Computed in INTEGFN, INTEGFP, INTEGFR
SFI	An approximation of the shape function for the source image of node i w.r.t. nodes j = 1,nt; $SFI = \sum_{i=1}^{nt} \sum_{m=1}^{mp} f_{iI+}^{k}$.	Computed in INTEGFN, INTEGFP, INTEGER
SFN	An approximation of the shape function for node i w.r.t nodes j = 1,nt; $SFN = \sum_{j=1}^{nt} \sum_{m=1}^{mp} f_{iN}^{k}$.	Computed in INTEGFN, INTEGFP, INGEGFR
theta	An orientation angle of the current fracture, nf, in radians;	Computed in ROFRAC

	theta = frac1(nf,2) * (pi/180).	
xc	The X-coordinate of center of current fracture, nf; xc = frac1(nf,5).	Set in ROFRAC
xii	The x-coordinate of the image source of a point node.	Computed in POINT
xl1 xl2	The lower and upper limits of integration, ξ_1 and ξ_2, of a radial node in the x' y' -coordinate system.	Computed in RADIAL
yc	The Y-coordinate of center of current fracture, nf; yc = frac1(nf,6).	Set in ROFRAC
yii	The y-coordinate of the image source of a point node.	Computed in POINT
yl1 yl2	The lower and upper limits of integration, ξ_1 and ξ_2, of a nonradial node in the x' y' -coordinate system	Computed in NRADIAL
zc	Z-coordinate of center of current fracture, nf; zc = frac1(nf,7).	Set in ROFRAC

Table C-2. Description of DISCEL Program Arrays

Variable	Description
Global:	
a(nonz)	Linear array containing the non-zero elements of the global mass balance matrix A, keyed by arrays ifirst, irlast,iclast,irnext,icnext. Filled in FULFIL and solved in CGSOLV, i = 1,nterm a(i) = value of term in entry i of matrix A.
bm(maxn)	One half the node widths for the nt nodes on the current fracture, nf, computed in ROFRAC.
BMM(maxnod,3)	The right hand side of global mass balance equations, mdir vectors, filled in FULFIL; after calling solver, contains solved average head values.
fluxb(6,3)	The total flux through the six boundary planes in mdir directions; computed in SFLUX.
G(maxn,maxn)	The matrix of shape factors, Kb [G], for current fracture, nf, relating nodal flux to average nodal head, computed in FORMG, stored in STOREG, and used again in SFLUX.
Gstore(maxn*maxfrc)	A linear array of fracture numbers and stored G matrices of fractures containing boundary nodes; the counter k (globally kstore), keeps track of the last position filled. When STOREG is called for a given fracture, nf, with nt nodes: - Gstore(k+1) = nf and the next nt*nt locations are filled with the G matrix of fracture nf; - k is reset to the original k + nt*nt +1; - the next location, Gstore(k+1), is set to maxfrc + 10 as a flag to mark the last stored G matrix. SFLUX retrieves the stored G matrices to compute flux after the global mass balance equations are solved.
head(maxh)	A linear array of constant average head values of boundary nodes; mdir values per node; keyed by global node number by keyh; computed in BNDCON and used in FULFIL.
iclast(maxnod)	Key to array a: number of last entry in array a for a column of matrix A, iclast(i), i=1, nnodes iclast(i) = number of last entry in array a for an element of column i in matrix A.

icnext(nonz) — Key to array a: number of next entry in array a for the same row in matrix A, i=1,nterm
icnext(i): number of next entry in array a for an element of the same row as entry i.

icol(nonz) — Column number in global mass balance matrix A of elements of array a, i=1,nterm.
icol(i) = column number of entry i in array a

ifirst(maxnod) — Key to array a, containing the entry number in array a of diagonal elements of matrix A,
i = 1,nnodes
ifirst(i) = entry number in array a of diagonal element (i,i) in matrix A.

irlast(maxnod) — Key to array a: number of last entry in array a for a row of matrix A, irlast(i), i=1,nnodes,
irlast(i) = number of last entry in array a for an element of row i in matrix A.

irnext(nonz) — Key to array a: number of next entry in array a for the same row in matrix a, i=1,nterm
irnext(i): number of next entry in array a for an element of the same row as entry i.

irow(nonz) — Row number in global mass balance matrix A of elements of array a, i=1,nterm
irow(i) = row number of entry i in array a

iside(3,2) — An array used to correlate parallel flow region boundary planes, set in a data statement in DISCEL, used in BNDCON:

$$\begin{bmatrix} 1 & 3 \\ 5 & 6 \\ 2 & 4 \end{bmatrix}$$

keyh(maxnod) — Key to head array by global node number; for node nn:
- keyh(nn) = starting location and
- keyh(nn) + mdir - 1 = ending location in array head of assigned head values in mdir directions;
set in BNDCON, and used in FULFIL.

ngnn(maxn) — Array of global node numbers corresponding to local node numbers 1 to nt on current fracture nf; set in ROFRAC and used in FULFIL. Set again in SFLUX and used locally.

ntype(maxn) — Node type of each of the nt nodes on current fracture, nf; set in ROFRAC and used in DISCEL:
ntype(i) = 1 if node i is a point node,
ntype(i) = 2 if node i is a radial node,
ntype(i) = 3 if node i is a nonradial node.

rl(maxn) — Node length of each of the nt nodes on currrent fracture, nf;

	computed in ROFRAC and used in NRADIAL, INTEGFN, and INTEGFR
scrptF(maxn,maxn)	Matrix of shape functions Υ_{ij}^{nf}, for current fracture, nf; computed in INTEGFN, INTEGFP, and INTEGFR and used in FORMG.
wkarea(maxn*(maxn+3))	Work area needed for library matrix inversion subroutine LINV2F.
x1(maxn) x2(maxn) y1(maxn) y2(maxn)	Coordinates of the endpoints of the nt nodes on the current fracture, nf, in the local 2-D coordinate system of the fracture; computed in ROFRAC and used in POINT, INTEGFP, RADIAL, INTEGFR, NRADIAL, INTEGFN.
Local	
asearch(maxnod)	Product of matrix A and vector search, computed and used at each iteration of the matrix solving subroutine CGSOLV.
fluxn(maxnod,3)	The total flux through each node w.r.t. the global system; should equal zero for internal nodes; computed in SFLUX.
fluxt(3)	The total flux through the six boundary planes in mdir directions; should equal zero; computed in SFLUX.
h(maxn,3)	The solved average heads for the nt nodes on the current fracture in mdir directions; set in SFLUX to appropriate values of hd array.
hd(maxnod,3)	The solved average heads for the nnode nodes of the fracture mesh in mdir directions; hd is the local name used in SFLUX for the solution vectors BMM(nnode,3).
Q(maxn,3)	The flux through each of the nt nodes on the current fracture w.r.t. the other nt-1 nodes on the fracture; computed in SFLUX.
res(maxnod)	residual vector, during iterations for matrix solving in subroutine CGSOLV.
rot(3,3)	The rotation matrix from the global XY-coordinate system to the local xy-coordinate system of the current fracture, nf; computed and used in ROFRAC.
scrptG(maxn,maxn)	The matrix of shape factors, Γ^{nf}, for the current fracture, nf; an intermediate step between scrpt F and G; computed and used in FORMG.
search(maxnod)	search direction vector during iterations for matrix solving, computed at each iteration, in subroutine CGSOLV.
sumQ(3)	The sum of the fluxes through the nt nodes on the current fracture in mdir directions; should equal zero; computed in SFLUX.

Table C-3. DISCEL Subroutine Outline

DISCEL		
	READ2	
	BNDCON	
	ROFRAC	
	POINT	INTEGFP
	RADIAL	INTEGFR
	NRADIAL	INTEGFN
	FORMG	LINV2F*
	STOREG	
	FULFIL	
	CGSOLV	
	SFLUX	

*IMSL Subroutine, (International Mathematical and Statistical Library).

Table C-4. Description of DISCEL Subroutines

Subroutine	Description
BNDCON	Reads boundary condition parameters and computes head values for boundary nodes. One, two or three flow directions may be used given: new,nnode,iside(3,2),nside(nnode,2), frcoef(6,4),node1(nnode,6),maxh reads: ibcc,ndir(3),delphi(3) or seth(6,maxd) returns: new,maxd,keyh(nnode),head(maxh)
CGSOLV	Matrix solver using the conjugate gradient method. given: nnode,ib,a(nonz),ifirst(nnode),irlast(nnode), iclast(nnode),irnext(nonz),icnext(nonz),bmm(nnode,maxd) returns: modified bmm(nnode,ib) containing the solution, i.e., the head value at each node in flow direction ib; ier, error flag.
FORMG	Forms the G matrix of the current fracture from scrptF matrix such that: Kb*[G]*[head] = [Q] given: maxn,nf,nt,scrptF(nt,nt),pkb, wkarea(maxn*(maxn+3)) returns: G(nt,nt) uses subroutine: LINV2F
FULFIL	Fills the array a with the non zero elements of the global mass balance matrix A and parts of the solution vector bmm, with entries for the current fracture's G matrix. given: maxnd, g(maxnd,maxnd), nt,ngnn(maxn), head(maxh), node2(maxnod,2), keyh(maxnod), a(nonz),bmm(nnode,maxd) returns: modified a, bmm
INTEGFN	Approximates integration of head shape functions sfn, sfi and sf0 over the nt nodes on fracture nf w.r.t. a non-radial node, i given: maxn,nf,i,nt,ninteg,calphx,cbetax,p,yl1,yl2, radius,x1(nt),y1(nt),x2(nt),y2(nt),rl(nt), bm(nt),ntype(nt) returns: scrptF(i,j) for j=1,nt
INTEGFP	Approximates integration of head shape functions sfn, sfi and sf0 over the nt nodes on fracture nf w.r.t. a point node, i given: maxn,nf,i,nt,ninteg,xii,yii,radius,x1(nt), y1(nt),x2(nt),y2(nt),r1(nt),bm(nt),ntype(nt) returns: scrptF(i,j) for j=1,nt
INTEGFR	Approximates integration of head shape functions sfn, sfi and sf0

over the nt nodes on fracture nt w.r.t. a radial node, i

given: maxn,nf,i,nt,ninteg,calphx,cbetax,xl1,xl2,
radius,x1(nt),y1(nt),x2(nt),y2(nt),rl(nt),
bm(nt),ntype(nt)
returns: scrptF(i,j) for j=1,nt

IROUND — round-off function

given: x,nplace
returns: iround=integer part of x*10**nplace

LINV2F — An IMSL* matrix inversion subroutine using full storage mode; used by subroutine FORMG to invert the modified script F matrix. Note, n = nt+1, where nt is the number of nodes on the current fracture

given: n,maxn,scrptF(n,n),wkarea(n*n+n*3),idgt
returns: scrptG(n,n) -- the inverse; ier -- error flag

NRADIAL — Generates rotation parameters and integration limits for computation of head shape functions for the non-radial node i; calls subroutine INTEGFN to compute script F.

given: nt,i,x1(i),y1(i),x2(i),y2(i),rl(i)
returns: calphx,cbetax,p,yl1,yl2,
and scrptF(i,j) for j=1,nt from INTEGFN
uses subroutine: INTEGFN

POINT — Determines the coordinates of the image of the point node i; calls subroutine INTEGFP to compute script F.

given: nt,i,x1(i),y1(i),radius
returns: xii,yii and scrptF(i,j) for j=1,nt from INTEGFP
uses subroutine: INTEGFP

RADIAL — Determines rotation parameters and integration limits for computation of head shape functions for the radial node i; calls subroutine INTEGFR to compute script F.

given: nt,i,x1(i),y1(i),x2(i),y2(i)
returns: calphx,cbetax,xl1,xl2, and scrptF(i,j) for
j=1,nt from INTEGFR
uses subroutine: INTEGFR

READ2 — Reads data written by fracture mesh generator program, FMG3D, from unit 2 including run identification, flow region parameters, fracture characteristics, and nodal arrays; sets flag ndata.

ndata=1 normal data set, run program,
ndata=2 normal end of data, stop,
ndata=3 incomplete data set, error stop.

returns: iray,idate,title(40),
xmesh,ymesh,zmesh,rophi,rothe,frcoef(6,4).
nfrac,frac1(nfrac,8),coefp(nfrac,4),
nnode,nseg,nsegb,node1(nnode,6),
nside(nnode,2),node2(nnode,2),node3(node3(1)),

nkey(nfrac,2),ndata

ROFRAC — For the current fracture, nf:
- sets radius equal to frac1(nf,3),
- computes transmissivity, equal to permeability times aperture,
- computes coordinate transformation parameters from the global to the local fracture system,
- renumbers all nodes on the fracture from one to nt and stores a catalog of corresponding global node numbers,
- sets flag for storing G matrix if the fracture intersects any flow region boundary planes,
- for each node on fracture nf
  - transforms endpoint coordinates to the local system,
  - computes the length,
  - computes the width in the plane of fracture nf,
  - determines the node type -- point, radial, or non-radial.

given: maxfrc,maxnod,maxn,nfrac,nnode,nt,rogomu,
frac1(nfrac,8),coefp(nfrac,4),nkey(nfrac,2),
node1(nnode,6),node2(nnode,2),node3(nnode*2+1)
returns: radius,pkb,nt,n1,n2,ngnn(nt),istore,
x1(nt),y1(nt),x2(nt),y2(nt),rl(nt),bm(nt),
ntype(nt)

SFLUX — Computes flux through nodes on fracture intersecting flow region boundary planes; sums flux through each of the boundary planes. The G(nt,nt) matrix of each such fracture is retrieved from the linear array Gstore; the head values, h(nt,maxd), of nodes on the fracture are taken from the solution of the global mass balance equations, head(nnode,maxd); [Q] = [G] * [h].

given: maxfrc,maxnod,maxn,nnode,nkey(maxfrc,2),
node2(nnode,2),node3(maxnod**2+1),maxd,
head(nnode,maxd),Gstore(maxfrc*maxn)
returns: fluxb(6,maxd)

STOREG — Stores G matrix of each fracture intersecting one or more flow region boundary planes for computation of flux after solution of global mass balance equations. The nt by nt G matrices are stored in the linear array Gstore; the counter k keeps track of the last location filled; for a given fracture, nf, the first location, k+1, stores the fracture number and the following nt**2 contain the G matrix.

given: maxfrc,maxn,nf,nt,G(nt,nt),k
returns: Gstore(maxfrc*maxn)

APPENDIX D

Listing of Program DISCEL

include files for DISCEL:

```
file COMFLR.BLKD
     common /floreg/xmesh,ymesh,zmesh,rophi,rothe,
    1        frcoef(6,4),iside(3,2)

file COMFND.BLKD
     common /fnodes/nt,n1,n2,ngnn(maxn),x1(maxn),y1(maxn),
    1 x2(maxn),y2(maxn),r1(maxn),bm(maxn),ntype(maxn),
    2 radius,istore

file COMFRC.BLKD
     common /fract/nfrac,frac1(maxfrc,8),coefp(maxfrc,4)

file COMHED.BLKD
     common /heads/head(maxh),keyh(maxnod),maxd,ibcc,
    1 ndir(3),delphi(3),seth(6,3)

file COMIDT.BLKD
     character iray*7,idate*9,nray*7,ndate*9
     character*4 ititle(40)
     common /ident/iray,idate,nray,ndate,ititle

file COMMAT.BLKD
     common /aandb/a(nonz),bmm(maxnod,3),res(maxnod),
    ,              search(maxnod),asearch(maxnod),
    ,              irow(nonz),icol(nonz),
    ,              ifirst(maxnod),irlast(maxnod),iclast(maxnod),
    ,              irnext(nonz),icnext(nonz)

file COMNOD.BLKD
     real node1(maxnod,6)
     common /nodes/nnode,nnfrac,node1,nside(maxnod,2),
    1 node2(maxnod,2),node3(maxnod*2+1),nkey(maxfrc,2),
    2 nseg,nsegb

file MAXDIM.BLKD
     parameter (maxfrc=1000,maxnod=7000,maxn=50,
    ,           maxh=1000,nonz=50000)
```

```
c
c
      PROGRAM DISCEL
c
c
c ************   P R O G R A M   D I S C E L   ************
c
c
c
c *******   THREE-DIMENSIONAL FRACTURE FLOW MODEL   *******
c
c                        written by
c
c **************   PEGGY (MACLEAN) GILMOUR   **************
c
c                           for
c
c ********************   JANE LONG   **********************
c
c
c
c
      include 'maxdim.blkd/list'
      parameter (mnod3=3*maxnod)
      include 'comfrc.blkd/list'
      include 'comflr.blkd/list'
      include 'comidt.blkd/list'
      include 'comnod.blkd/list'
      include 'comhed.blkd/list'
      include 'comfnd.blkd/list'
c     include 'comABM.blkd/list'
      include 'commat.blkd/list'
c
      dimension scrptF(maxn,maxn),G(maxn,maxn),Gstore(maxn*maxfrc)
      equivalence (scrptF,G)
      dimension fluxb(6,3)
      dimension wkarea(maxn*(maxn+3))
      character*3  units(2)/'cgs','mks'/
      character*6 borf(2)/'banded','full  '/
      data iside/1,5,2,3,6,4/
      data nray/'DISCEL '/
      data BMM/mnod3*0.0/
c
c
c
      maxnd=maxn
      mfrac=maxfrc
c
c *** Request job information.
      call date(ndate)
c
c *** Read control parameters.
      read(1,105) iunits,ninteg
c
c *** Set units: iunits=1 -- cgs, iunits=2 -- mks.
      rogomu = 8172.16667
      if (iunits .eq. 2) rogomu = 817.216667
c
c *** Print title page.
      write (3,110)
```

```
c
c *** Read mesh generated by FMG3D.
      inew=1
      nmesh=0
    5 read (1,106,end=10) newbc
      new=newbc
      go to 12
   10 inew=2
   11 new=0
   12 nmesh=nmesh+1
      nbcs=0
      call READ2(ndata)
      go to (15,90,92), ndata
   15 if ((nfrac .gt. maxfrc) .or. (nnode .gt. maxnod)) go to 94
c
c *** Print mesh and run parameters.
      write (3,112) nmesh
      write (3,115) nray,ndate,iray,idate
      write (3,117) (ititle(i),i=1,40)
      write (3,120) units(iunits),ninteg
      write (3,127) xmesh,ymesh,zmesh,rophi,rothe
      write (3,125) nnode,maxnod,nfrac,maxfrc,nseg,nsegb
c
c *** Compute bandwidth
c     call BWIDTH(ibw,kdiag,kfill,matrxsz)
c     if (matrxsz .gt. mnodsq) go to 96
c     write (3,122) ibw,borf(kfill),matrxsz
c
c *** Read boundary conditions and compute head values for each node.
   20 nbcs=nbcs+1
      write (3,130) nmesh,nbcs
      call BNDCON (new)
c
c
c
c ******************  Begin fracture loop  **************
c
      write (3,135)
      mdir=maxd
      mnode=nnode
      kstore = 0
      nterm=0
      do nfr=1,nfrac
        nf=nfr
c
c ***   Call subroutine ROFRAC to transform all nodes on fracture nf to
c ***   the local coordinate system of that fracture.  A local numbering
c ***   system is set up for nodes 1 to nt.  The length, width, and type
c ***   of each node on fracture nf are determined.
c ***   Fracture transmissivity, permeability * aperture, is computed.
c
        call ROFRAC(nf,rogomu,pkb)
        if (nt .gt. maxn) go to 98
        nnpf=nt
c
c **********  Begin loop on nodes on fracture nf ********
c
        do inode=1,nt
          i=inode
c
```

```
c ***     Calculate coefficients of shape functions for node i and
c ***     scrptF(j,i) of nodes j=1,nt on node i by integration of FN
c ***     and FI for each node j on fracture nf.
c
          if(ntype(i).eq.1)then
            call POINT(maxnd,i,ninteg,scrptF)
           elseif(ntype(i).eq.2)then
            call RADIAL(maxnd,i,ninteg,scrptF)
           elseif(ntype(i).eq.3)then
            call NRADIAL(maxnd,i,ninteg,scrptF)
           endif
c
c **********  End loop on nodes on fracture nf  *********
c
         enddo
c
c ***   Call subroutine FORMG to compute G matrix for fracture nf.
c
        call FORMG(maxnd,nf,nnpf,scrptF,G,pkb,wkarea)
c
c ***   If fracture nf intersects one or more boundaries, store G(i,j).
c
        if (istore .eq. 1) call STOREG(mfrac,maxnd,nf,nnpf,G,
     ,                                 Gstore,kstore)
c
c       fill array a with only nonzero elements of big matrix,
c       and solution vector.
c
        call fulfil(maxnd,g)
c
c **************  End fracture loop  *************
c
       enddo
c
c
c ******  Solve Global Mass Balance Equations and Flux  ******
c     call iterative algorithm (Conjugate Gradient) for each
c     set of boundary conditions
c
      do ib=1,maxd
        call cgsolv(ib,ier,mnode)
        if (ier.eq.1)stop
       enddo
c
c *** Call SFLUX to back calculate fluxes through the boundary nodes.
c
      call SFLUX(Gstore,mdir,fluxb)
c
c
c
c *** Zero BMM array for next data set.
c
      do j=1,3
        do i=1,maxnod
          BMM(i,j)=0.0
         enddo
       enddo
c
c *** Go to 5 or 12 to read next mesh or
c *** go to 20 for next set of boundary conditions.
```

```
      if (new .gt. 0) go to 20
      go to (5,11),inew
c
c *** Normal stop at end of data ***
c
   90 write (3,190)
      stop
c
c ******** ERROR STOPS **********
c
c *** Stop, incomplete data set from FMG3D.
   92 write (3,191)
      write (3,192)
      stop
c
c *** Stop, maximum dimensions exceeded.
   94 write (3,193)
      write (3,194) nfrac,maxfrc,nnode,maxnod
      stop
c
   98 write (3,193)
      write (3,198) nt,nf,maxn
      stop
c
c
c ***************************************************************************
c
c ************************   FORMATS   ***********************
c
  105 format (2(10x,i5))
  106 format (10x,i5)
  110 format(1h1,5(/),10x,10(6h******),///,23x,12hT H R E E - ,
     1 22hD I M E N S I O N A L ,///,22x,16hF R A C T U R E ,
     2 22h  F L O W   M O D E L ,///34x,12hD I S C E L ,///,
     3 30x,20hw r i t t e n   b y ,///,31x,18hJ A N E  L O N G ,
     4 ///,37x,6ha n d ,///,17x,30hP E G G Y  ( M A C L E A N ) ,
     5 16h  G I L M O U R ,///,10x,10(6h******),16(/))
  112 format (1h1,10(1h*,1x),11hMESH NUMBER,i5,10(1x,1h*),///)
  115 format (1h0,8hPROGRAM ,a7,5h --  ,a9,/,
     1 25h using mesh generated by ,a7,5h --  ,a9,/)
  117 format (1h ,20a4)
  120 format (1h0,10hunits are ,a3,/,
     1 31h number of integration steps = ,i5,/)
  122  format (1h0,13h bandwidth = ,i5,17h, therefore, use ,a6,
     1 14h matrix solver,/,17h,  matrix size = ,i6)
  125 format (1h0,8h nnode =,i5,10h, maxnod =,i5,/,
     1        1h0,8h nfrac =,i5,10h, maxfrc =,i5,/,
     2        1h0,8h nseg  =,i5,10h   nsegb =,i5)
  127 format (1h0,20h *** FLOW REGION ***,//,9h  xmesh =,f10.4,
     1  10h   ymesh =,f10.4,10h   zmesh =, f10.4,/,
     2    9h  rophi =,f10.4,10h   rothe =,f10.4,//)
  130 format (1h1,10(1h*,1x),11hMESH NUMBER,i5,10(1x,1h*),//,
     1 1h0,7(1h*,1x),23hBOUNDARY CONDITION SET ,i5,7(1x,1h*),///)
  135 format (1h1,31h *** NODES ON EACH FRACTURE ***,/)
  190 format (1h0,27(1h*),//,27h Normal stop at end of data)
  191 format (1h0,24h****** ERROR STOP ******)
  192 format  (1h0,31(1h*),//,31h Incomplete data set from FMG3D)
  193 format (1h0,24h****** ERROR STOP ******,///,
     1      41h****** MAXIMUM DIMENSIONS EXCEEDED ******)
  194 format (1h0,23h    number of fractures,i5,
```

```
     1 16h exceeds maximum,i5,//,
     2 19h or number of nodes,i5,
     3 16h exceeds maximum,i5)
  198 format (1h0,17h  number of nodes,i5,12h on fracture,i5,
     1  16h exceeds maximum,i5)
c
  200 format (3f10.4)
c
      end
c
c
c
c *** ROUND-OFF FUNCTION
c
      function iround(a,nplace)
      aplace=a*10.**nplace
      iround=nint(aplace)
      return
      end
      SUBROUTINE BNDCON(new)
c
c *** If boundary condition option ibcc =1 :
c *** subroutine bndcon reads boundary conditions in the form of flow
c *** direction and delta phi, for one, two, or three directions,
c *** and calculates head values for all boundary nodes.  Flow
c *** directions are determined by the value of ndir(i), i=1,3:
c ***   ndir(i) = 1   flow in x direction from side 1 to 3,
c ***   ndir(i) = 2   flow in y direction from side 5 to 6,
c ***   ndir(i) = 3   flow in z direction from side 2 to 4.
c *** The value of head on the inflow side is delta phi and on the
c *** outflow side head is zero.  The direction of flow can be reversed
c *** by using a negative value of delta phi.
c
c *** If ibbc = 2: head values are read in for each flow region
c *** boundary plane and head values set for nodes intersecting
c *** those planes.  Up to three sets of head values can be set.
c
      include 'maxdim.blkd/list'
      include 'comflr.blkd/list'
      include 'comnod.blkd/list'
      include 'comhed.blkd/list'
c
      dimension hd(2)
      character*55 btype(2)/
     1' Constant average gradient for directional permeability',
     2' Set head values at each boundary plane               '/
c
c
c *** Read boundary condition option and flow directions,
c *** set the number of flow directions.
c
      if (new .le. 0) go to 7
    5 read (1,110,end=93) ibcc
      if (ibcc .gt. 2) go to 95
      read (1,120,end=93) (ndir(i),i=1,3)
      maxd=0
      if (ndir(1) .ne. 0) maxd=1
      if (ndir(2) .ne. 0) maxd=2
      if (ndir(3) .ne. 0) maxd=3
      if (maxd .eq. 0) go to 97
```

```
    7 write (3,200) btype(ibcc),maxd
c
c *** Zero keyh array, if keyh(nn)=0 then node nn is not a boundary node.
      do 10 nn=1,nnode
   10 keyh(nn)=0
c
      go to (15,65),ibcc
c
c *** Read delta phis.
   15 if (new .gt. 0) read (1,130,end=93) (delphi(i),i=1,3)
      write (3,210)
c
      do 60 k=1,maxd
      jk=k
      nin =iside(ndir(k),1)
      nout=iside(ndir(k),2)
      dmesh=2.0*frcoef(nout,4)
      write (3,220) k,nin,nout,delphi(k)
c
      do 50 nn=1,nnode
      if (node2(nn,2) .gt. 0) go to 50
c
      np=-3
      do 30 npt=1,2
      np=np+3
      hd(npt)=0.
      ns=nside(nn,npt)
      if (ns .eq. nout) go to 30
      if (ns .ne. nin) go to 20
      hd(npt)=delphi(k)
      go to 30
c
c *** Compute head for a point on a wedge shaped boundary.
   20 d=0.0
      do 25 i=1,3
   25 d=d + frcoef(nout,i)*node1(nn,i+np)
      d=abs(d + frcoef(nout,4))
      hd(npt)=(d*delphi(k))/dmesh
c
   30 continue
c
c *** Average heads at endpoints 1 and 2 and key location of head of
c *** node nn.  If the node has one boundary endpoint and one interior
c *** endpoint then half the boundary endpoint head is stored for fillAB.
      head(jk)=(hd(1)+hd(2))*0.5
      go to (35,40,40),k
   35 keyh(nn)=jk
   40 jk=jk+maxd
   50 continue
c
   60 continue
c
c
      go to 90
c
c
c ****************************************************************
c
c
c *** Read head values for each boundary plane in maxd directions.
```

```
c
   65 if (new .gt. 0) read (1,160,end=93)
     1                     ((seth(i,j),i=1,6),j=1,maxd)
      write (3,230)
      write (3,240) (j,(seth(i,j),i=1,6),j=1,maxd)
      do 80 k=1,maxd
      jk=k
      do 75 nn=1,nnode
      if (node2(nn,2) .gt. 0) go to 75
c
      ns=nside(nn,1)
      head(jk)=seth(ns,k)
c
c *** Key location of head for node nn.
      go to (70,72,72),k
   70 keyh(nn)=jk
   72 jk=jk+maxd
   75 continue
c
   80 continue
c
c
c
c
   90 nhead=jk-maxd
      write (3,180) nhead,maxh
      if (nhead .gt. maxh) go to 98
      new=new-1
c
      return
c
c
c
c *** Write error messages and stop the program.
c
   93 write (3,140)
      write (3,135)
      stop
c
   95 write (3,140)
      write (3,145)
      stop
c
   97 write (3,140)
      write (3,150)
      stop
c
   98 write (3,140)
      write (3,170)
      stop
c
  110 format (10x,i5)
  120 format (10x,3i5)
  130 format (10x,3f10.4)
  135 format (36h incomplete boundary condition input)
  140 format (1h0,36h ***ERROR STOP *** subroutine BNDCON)
  145 format (12h ibcc .gt. 2)
  150 format (30h no flow directions were input)
  160 format (10x,6f10.4)
  170 format (45h number of head values exceeds dimension maxh)
```

```
  180 format (1h0,8h nhead =,i5,10h, maxh   =,i5,/)
  200 format (1h0,28h *** BOUNDARY CONDITIONS ***,//,a55,1h,,/,
     1  20h flow is computed in,i2,12h directions.,/)
  210 format (1h ,36h  flow      inflow   outflow   delta,/,
     1           36h direction    side     side      phi,/)
  220 format (i6,i11,i9,f11.4)
  230 format (1h ,35h  flow     heads at boundary planes,/,
     1 11h  direction,4x,1h1,9x,1h2,9x,1h3,9x,1h4,9x,1h5,9x,1h6,/)
  240 format (i7,1x,6f10.4)
c
      end
c
c
      subroutine cgsolv(ib,ier,mnode)
c
c     solves a*x=bmm for a symmetric definite positive matrix
c     by the conjugate gradient method.
c
      include 'maxdim.blkd'
      include 'commat.blkd'
c
      parameter (prec=1.0e-8)
c
c     initialise arrays
c
      ier=0
      rtrold=0.
      do i=1,mnode
        res(i)=bmm(i,ib)
        k=ifirst(i)
        do while(k.gt.0)
          res(i)=res(i)-a(k)*0.5
          k=irnext(k)
         enddo
        k=ifirst(i)
        k=icnext(k)
        do while(k.gt.0)
          res(i)=res(i)-a(k)*0.5
          k=icnext(k)
         enddo
        search(i)=res(i)
        bmm(i,ib)=0.5
        rtrold=rtrold+res(i)*res(i)
       enddo
c
c     begin iterations
c
      do iter=1,1000
c
c       compute new asearch
c
        do i=1,mnode
          asearch(i)=0.
          k=ifirst(i)
          do while(k.gt.0)
            j=icol(k)
            asearch(i)=asearch(i)+a(k)*search(j)
            k=irnext(k)
           enddo
          k=ifirst(i)
```

```
            k=icnext(k)
            do while(k.gt.0)
              j=irow(k)
              asearch(i)=asearch(i)+a(k)*search(j)
              k=icnext(k)
             enddo
           enddo
c
c       compute alpha
c
          alden=0.
          do i=1,mnode
            alden=alden+search(i)*asearch(i)
           enddo
          alpha=rtrold/alden
c
c       compute new bmm , res and rtr and error
c
          rtr=0.
          flagpr=0
          error=0.
          do i=1,mnode
            as=alpha*search(i)
            bmm(i,ib)=bmm(i,ib)+as
            res(i)=res(i)-alpha*asearch(i)
            rtr=rtr+res(i)*res(i)
            error=error+as*as
            if(abs(as).gt.prec)flagpr=1
           enddo
c
c       if precision attained, return
c
          error=sqrt(error)/mnode
c         if(error.le.prec)return
          if(flagpr.eq.0)return
c
          beta=rtr/rtrold
          rtrold=rtr
          do i=1,mnode
            search(i)=res(i)+beta*search(i)
           enddo
         enddo
c
c       if precision could not be reached, write
c       error message
c
        write(3,100)error
        ier=1
        return
  100   format(' precision reached after 1000 iterations'/
     '       ' of cgsolv is only ',g10.4)
        end

c
c
        SUBROUTINE FORMG(mn,nf,nt,scrptF,G,pkb,wkarea)
c
c
c
        include 'maxdim.blkd'
```

```
c
      dimension scrptF(mn,mn),G(mn,mn)
      dimension scrptG(maxn,maxn)
      dimension wkarea(1)
c
c
      n=nt+1
c
c *** add a column and a row of ones to scrptF
c
      do i=1,nt
        scrptF(i,n)=1.
        scrptF(n,i)=1.
       enddo
      scrptF(n,n)=0.
c
c     take average of scrptf(i,j) and scrptf(j,i), to
c     get rid of numerical error in shape factor calculation.
c
      do i=1,nt-1
        do j=i+1,nt
          scrptf(i,j)=(scrptf(i,j)+scrptf(j,i))/2.
          scrptf(j,i)=scrptf(i,j)
         enddo
       enddo
c
c *** Invert scrptF to get scrptG.
c
      idgt=0
      call LINV2F(scrptF,n,maxn,scrptG,idgt,wkarea,ier)
      if (ier .eq. 129 .or. ier .eq. 131)then
        write (3,100) ier,nf
        stop
       endif
c
c *** multiply by fracture transmissivity, pkb, to get G.
c
      do i=1,nt
        do j=i,nt
          g(i,j)= pkb *(scrptg(i,j)+scrptg(j,i))/2.
          g(j,i)=g(i,j)
         enddo
       enddo
c
      return
c
c
  100 format (1h0,34h *** matrix inversion error, ier =,i5,
     1        10h, fracture,i5,/,23h ****  ERROR STOP  ****,
     2        17h subroutine FORMG)
c
      end
c
      SUBROUTINE fulfil(mn,G)
c
c *** This subroutine fills the array a with the non zero elements
c *** of the global mass balance matrix and parts of the solution
c *** vector BMM.  The head array contains the non-zero elements
c *** of the boundary condition vector.  Arrays head and BMM contain
c *** values for flow in one, two, or three directions depending on
```

```
c *** maxd.  nj and ni are local node numbers, j and i are
c *** corresponding global node numbers.
c
      include 'maxdim.blkd/list'
      include 'comnod.blkd/list'
      include 'comhed.blkd/list'
      include 'comfnd.blkd/list'
      include 'commat.blkd/list'
c
      dimension G(mn,mn)
c
c     fill the diagonal elements
c
      do nj=1,nt
        j=ngnn(nj)
        if(ifirst(j).eq.0)then
          nterm=nterm+1
          ifirst(j)=nterm
          if (node2(j,2) .ge. 0)then
            a(nterm)=g(nj,nj)
           else
            a(nterm)=1.0
            do ib=1,maxd
              khj=keyh(j)+ib-1
              bmm(j,ib)=head(khj)
             enddo
           endif
          irnext(nterm)=0
          icnext(nterm)=0
          irow(nterm)=j
          icol(nterm)=j
          irlast(j)=nterm
          iclast(j)=nterm
         elseif(node2(j,2).ge.0)then
          ifirsj=ifirst(j)
          a(ifirsj)=a(ifirsj)+g(nj,nj)
         endif
       enddo
c
c     fill non-diagonal elements
c
      do nj=1,nt-1
        j=ngnn(nj)
        do ni=nj+1,nt
          i=ngnn(ni)
          if(node2(j,2).ge.0.and.node2(i,2).ge.0)then
c
c           i and j interior nodes: fill a
c
            nterm=nterm+1
            a(nterm)=g(nj,ni)
            last=irlast(j)
            irnext(last)=nterm
            last=iclast(i)
            icnext(last)=nterm
            icol(nterm)=i
            irow(nterm)=j
            irlast(j)=nterm
            iclast(i)=nterm
            irnext(nterm)=0
```

```
            icnext(nterm)=0
           else
            if(node2(j,2)*node2(i,2).le.0)then
c
c             i or j interior node: fill bmm
c
              if(node2(j,2).ge.0)then
                ii=i
                jj=j
               else
                ii=j
                jj=i
               endif
              do ib=1,maxd
                khii=keyh(ii)+ib-1
                bmm(jj,ib)=bmm(jj,ib)-g(nj,ni)*head(khii)
               enddo
             endif
           endif
         enddo
       enddo
      return
      end
c
c
      SUBROUTINE INTEGFN (mn,i,ninteg,calphx,cbetax,
     1                    yl1,yl2,p,scrptF)
c
c
c *** This subroutine approximates integration of FN and FI over
c *** the nodes on fracture nf when node i is a non-radial node.
c
c
      include 'maxdim.blkd/list'
      parameter (zerock=0.000005)
      parameter (pi4co=-0.0795774715,ylnmin=1.0000001)
      include 'comfnd.blkd/list'
c
      dimension scrptF(mn,mn)
c
c
c *** Compute terms independent of node j.
c
      rli=rl(i)
      coef=pi4co/rli
      yl12=2.0*yl1
      yl22=2.0*yl2
      yl1sq=yl1*yl1
      yl2sq=yl2*yl2
      p2=2.0*p
      psq=p*p
      a2=radius*radius
      a4=a2*a2
      a22=2.0*a2
      asqp2=a22*p
      y2myl2=yl22-yl12
      fiterm=yl1*log(yl1sq+psq)-yl2*log(yl2sq+psq)+
     1       p2*(atan(yl1/p)-atan(yl2/p))
      fi2=fiterm + y2myl2*(log(a2)+1.0)
c
```

```
c
c *** Calculate head shape function SFN for node i on itself.
c
      rlisq=rli*rli
      if (bm(i) .gt. 0.) go to 10
c
c *** Compute SFN for boundary node on itself.
c
      SFNii=(log(rlisq)-3.0)*rli*ninteg
      go to 15
c
c *** Compute SFN for interior node on itself.
c
   10 bmisq=bm(i)*bm(i)
      blinv=rli/bm(i)
      blsq=bmisq/rlisq
      SFNii=((1.0-blsq)*log(rlisq+bmisq)+blsq*log(bmisq)-3.0+
     1      (4.0/blinv)*atan(blinv))*rli*ninteg
c
c
c *** Begin loop on nodes j=1,nt.
c
   15 do 60 j=1,nt
c
      SFN=0.0
      SFI=0.0
      SF0=0.0
      nin=ninteg
      ij=1
      if (j .ne. i) go to 20
c
c *** Set parameters when j = i.
c
      ij=2
      SFN=SFNii
      xjn=p
      y1p=y11
      ylnin=rli/ninteg
      go to 25
c
c *** Rotate endpoints of node j to local x'y' system of node i.
c
   20 x1p= calphx*x1(j)+cbetax*y1(j)
      y1p= calphx*y1(j)-cbetax*x1(j)
      x2p= calphx*x2(j)+cbetax*y2(j)
      y2p= calphx*y2(j)-cbetax*x2(j)
c
      go to (22,21,21),ntype(j)
   21 xlnin=(x2p-x1p)/ninteg
      ylnin=(y2p-y1p)/ninteg
      go to 25
c
c *** Node j is a point source.
c
   22 nin=1
      xlnin=0.0
      ylnin=0.0
c
c *** Calculate head shape functions for node i and approximate
c *** integrating that function over the length of node j.
```

```
c
   25 do 50 n=1,nin
c
c *** Determine integration intervals for SFN and SFI.
c
      yjn=y1p+(n-0.5)*ylnin
      go to (27,40),ij
   27 xjn=x1p+(n-0.5)*xlnin
c
c *** Compute SFN for node j on node i, j .ne. i.
c
   30 yl1my=yl1-yjn
      yl2my=yl2-yjn
      yl1mys=yl1my*yl1my
      yl2mys=yl2my*yl2my
      abxmp=abs(xjn-p)
      xmpsq=abxmp*abxmp
      abxmp2=2.0*abxmp
      if (abxmp .gt. zerock) go to 35
      abxmp2=0.0
      abxmp=0.001
      if (yl1mys .gt. zerock) go to 32
      yl1mys=ylnmin
      yl1my=0.0
      go to 35
c
   32 if (yl2mys .gt. zerock) go to 35
      yl2mys=ylnmin
      yl2my=0.0
c
c
   35 SFN=SFN+yl2my*log(xmpsq+yl2mys)+abxmp2*atan(yl2my/abxmp)
     1       -yl1my*log(xmpsq+yl1mys)-abxmp2*atan(yl1my/abxmp)
     2       -y2my12
c
c *** Compute SFI for node j on node i, j .eq. or .ne. i.
c
   40 ysq=yjn*yjn
      gamma=xjn*xjn+ysq
      if (gamma .lt. zerock) go to 46
      alpha=psq*gamma-asqp2*xjn+a4
      beta =-a22*yjn
      bet2g=0.5*beta/gamma
      q=abs(p2*gamma-a22*xjn)
      if (q .gt. zerock) go to 42
      qog=0.0
      q=0.001
      go to 44
c
   42 qog=q/gamma
c
   44 omega1=gamma*yl1sq+beta*yl1+alpha
      omega2=gamma*yl2sq+beta*yl2+alpha
      if (omega1 .lt. zerock) omega1=ylnmin
      if (omega2 .lt. zerock) omega2=ylnmin
c
c
      SFI= SFI + fiterm +
     1  (yl2+bet2g)*log(omega2)+qog*atan((gamma*yl22+beta)/q)
     2 -(yl1+bet2g)*log(omega1)-qog*atan((gamma*yl12+beta)/q)
```

```
      go to 49
c
c
c *** Compute SFI for x and y equal zero.
c
   46 SFI=SFI + fi2
c
c
c     compute SF0
c
   49 r0sq=xjn*xjn+yjn*yjn
      if(r0sq.lt.zerock)r0sq=ylnmin
      SF0=SF0+log(r0sq)
c
   50 continue
c
c *** Store (SFN + SFI) for node j on node i in scrptF(j,i).
c *** Note, this is the transpose of scritp F.
c
      scrptF(j,i)=(SFN+SFI-SF0*rli)*coef/nin
c
c
c *** End loop on node j.
   60 continue
c
      return
c
      end
      SUBROUTINE INTEGFP(mn,i,ninteg,xii,yii,scrptF)
c
c
c *** This subroutine integrates FN and FI over the nodes on fracture
c *** nf when node i is a point source.
c
      include 'maxdim.blkd/list'
      parameter (zerock=0.000005)
      parameter (pi4co=-0.0795774715,alnmin=1.0000001)
      include 'comfnd.blkd/list'
c
      dimension scrptF(mn,mn)
c
c
      coef=pi4co
c
c *** Begin loop on nodes j=1,nt.
c
      do 60 j=1,nt
c
      if (j .eq. i) go to 15
      SFN=0.0
      SFI=0.0
      SF0=0.0
      nin=ninteg
      go to (22,21,21),ntype(j)
c
c *** Calculate head shape functions SFN for node i on itself and
c *** SFI for image of node i on itself when j=i.
c
   15 nin=1
      bmi=alnmin
```

```
      if (bm(i) .gt. 0) bmi=bm(i)
      SFN=log(bmi*bmi)
      gimgsq=(x1(i)-xii)**2+(y1(i)-yii)**2
      xylsq=x1(i)*x1(i)+y1(i)*y1(i)
      if (gimgsq .lt. zerock) gimgsq=alnmin
      if (xylsq .lt. zerock) xylsq=alnmin
      SFI=log(gimgsq)
      SF0=log(xylsq)
      go to 55
c
   21 xlnin=(x2(j)-x1(j))/ninteg
      ylnin=(y2(j)-y1(j))/ninteg
      go to 25
c
c *** Node j is a point source.
c
   22 nin=1
      xlnin=0.0
      ylnin=0.0
c
c *** Calculate head shape functions for node i and approximate
c *** integrating that function over the length of node j.
c
   25 do 50 n=1,nin
c
c *** Determine integration intervals for SFN and SFI.
c
      xjn=x1(j)+(n-0.5)*xlnin
      yjn=y1(j)+(n-0.5)*ylnin
c
c *** Compute SFN and SFI for node j on node i, j .ne. i.
c
      rnsq=(xjn-x1(i))**2+(yjn-y1(i))**2
      risq=(xjn-xii)**2+(yjn-yii)**2
      r0sq=xjn*xjn+yjn*yjn
      if (rnsq .lt. zerock) rnsq=alnmin
      if (risq .lt. zerock) risq=alnmin
      if (r0sq .lt. zerock) r0sq=alnmin
      SFN=SFN + log(rnsq)
      SFI=SFI + log(risq)
      SF0=SF0 + log(r0sq)
c
   50 continue
c
c *** Store (SFN + SFI) for node j on node i in scrptF(j,i).
c *** Note, this is the transpose of script F.
c
   55 scrptF(j,i)=(SFN + SFI -SF0)*coef/nin
c
c
c *** End loop on node j.
   60 continue
c
      return
c
      end
      SUBROUTINE INTEGFR(mn,i,ninteg,calphx,cbetax,xl1,xl2,scrptF)
c
c
c *** This subroutine integrates FN and FI over the nodes on fracture
```

```
c *** nf when node i is a radial node.
c
c
      include 'maxdim.blkd/list'
      parameter (zerock=0.000005)
      parameter (pi4co=-0.0795774715,xlnmin=1.0000001)
      include 'comfnd.blkd/list'
c
      dimension scrptF(mn,mn)
c
c
c *** Compute terms independent of node j.
c
      rli=rl(i)
      coef=pi4co/rli
      xl1sq=xl1*xl1
      xl2sq=xl2*xl2
      xl12=2.0*xl1
      xl22=2.0*xl2
      a2=radius*radius
      a4=a2*a2
      fiterm=-xl2*log(xl2sq)
      if (iround(xl1,4) .eq.0) go to 5
      fiterm=fiterm+xl1*log(xl1sq)
    5 fi2=fiterm+(xl2-xl1)*(log(a4)+2.0)
c
c
c *** Calculate head shape function SFN for node i on itself.
c
      rlisq=rli*rli
      if (bm(i) . gt. 0.) go to 10
c
c *** Compute SFN for boundary node on itself.
c
      SFNii=(log(rlisq)-3.0)*rli*ninteg
      go to 15
c
c *** Compute SFN for interior node on itself.
c
   10 bmisq=bm(i)*bm(i)
      blinv=rli/bm(i)
      blsq=bmisq/rlisq
      SFNii=((1.0-blsq)*log(rlisq+bmisq)+blsq*log(bmisq)-3.0+
     1        (4.0/blinv)*atan(blinv))*rli*ninteg
c
c
c *** Begin loop on nodes j=1,nt.
c
   15 do 60 j=1,nt
c
      SFN=0.0
      SFI=0.0
      SF0=0.0
      nin=ninteg
      ij=1
      if (j .ne. i) go to 20
c
c *** Set parameters when j=i.
c
      ij=2
```

```
      SFN=SFNii
      x1p=x11
      xlnin=rli/ninteg
      ysq=0.0
      aby2=0.0
      aby=0.001
      ny0=2
      go to 25
c
c *** Transform endpoints of node j to local x'y' system of node i.
c
   20 x1p= calphx*x1(j)+cbetax*y1(j)
      y1p=-cbetax*x1(j)+calphx*y1(j)
      x2p= calphx*x2(j)+cbetax*y2(j)
      y2p=-cbetax*x2(j)+calphx*y2(j)
      go to (22,21,21),ntype(j)
c
   21 xlnin=(x2p-x1p)/ninteg
      ylnin=(y2p-y1p)/ninteg
      go to 25
c
c *** Node j is a point source.
c
   22 nin=1
      xlnin=0.0
      ylnin=0.0
c
c *** Calculate head shape functions for node i and approximate
c *** integrating that function over the length of node j.
c
   25 do 50 n=1,nin
c
c *** Determine integration intervals for SFN and SFI.
c
      xjn=x1p+(n-0.5)*xlnin
      go to (27,40),ij
c
   27 yjn=y1p+(n-0.5)*ylnin
      ysq=yjn*yjn
      aby=abs(yjn)
      aby2=2.0*aby
      ny0=1
      if (aby .gt. zerock) go to 30
      aby2=0.0
      aby=0.001
      ny0=2
c
c *** Compute SFN for node j on node i, j .ne. i.
c
   30 x11mx=x11-xjn
      x12mx=x12-xjn
      x11mxs=x11mx*x11mx
      x12mxs=x12mx*x12mx
      go to (34,32),ny0
c
   32 if (x11mxs .lt. zerock) x11mxs=xlnmin
      if (x12mxs .lt. zerock) x12mxs=xlnmin
c
   34 SFN=SFN+x12mx*log(x12mxs+ysq)-x122+aby2*atan(x12mx/aby)
     1       -x11mx*log(x11mxs+ysq)+x112-aby2*atan(x11mx/aby)
```

```
c
c *** Compute SFI for node j on node i, j .eq. or .ne. i.
c
   40 asqx=a2*xjn
      asqy=a2*aby
      asqx2=2.0*asqx
      xsqysq=xjn*xjn+ysq
      go to (44,42),ny0
c
   42 if (xsqysq .lt. zerock) go to 46
c
   44 asqxxy=asqx/xsqysq
      asqyxy=a2*aby2/xsqysq
      omega1=a4-xl1*asqx2+xl1sq*xsqysq
      omega2=a4-xl2*asqx2+xl2sq*xsqysq
      if (omega1 .lt. zerock) omega1=xlnmin
      if (omega2 .lt. zerock) omega2=xlnmin
      xl1tan=(xsqysq*xl1-asqx)/asqy
      xl2tan=(xsqysq*xl2-asqx)/asqy
c
      SFI=SFI+fiterm
     1       +(xl2-asqxxy)*log(omega2)+asqyxy*atan(xl2tan)
     2       -(xl1-asqxxy)*log(omega1)-asqyxy*atan(xl1tan)
c
      go to 49
c
c *** Compute SFI for x and y equal zero.
c
   46 SFI=SFI+fi2
c
c     compute SF0
c
   49 r0sq=xjn*xjn+aby*aby
      if(r0sq.lt.zerock)r0sq=xlnmin
      SF0=SF0+log(r0sq)
c
   50 continue
c
c *** Store (SFN + SFI) for node j on node i in scrptF(j,i).
c *** Note, this is the transpose of script F.
c
      scrptF(j,i)=(SFN+SFI-SF0*rli)*coef/nin
c
c
c
c *** End loop on node j.
   60 continue
c
      return
c
      end
      SUBROUTINE NRADIAL(mn,i,ninteg,scrptF)
c
c
c *** This subroutine generates the rotation from the local
c *** system of fracture nf to the x'y' system of the non-radial
c *** node i such that the positive x'-axis is perpendicular
c *** to the line containing the node and the y'-axis is
c *** parallel to the node.
c *** The integration subroutine INTEGFN is called to compute
```

```
c *** script F for all nodes on fracture nf w.r.t. node i.
c
c
      include 'maxdim.blkd/list'
      include 'comfnd.blkd/list'
c
      dimension scrptF(mn,mn)

c
c
c *** The equation of the line connecting (x1,y1) and (x2,y2), in
c *** normal form, is: x cos(alpha) + y cos(beta) - p = 0, where
c *** p is the positive distance from (0,0) to the line.
c
      p=y1(i)*x2(i)-y2(i)*x1(i)
      xyl=rl(i)
      if (p .lt. 0.) xyl=-xyl
      p=p/xyl
c
      calphx=(y1(i)-y2(i))/xyl
      cbetax=(x2(i)-x1(i))/xyl
c
c
c *** Rotate the endpoints of node i to the x'y' system.
c *** This condenses to x1p = p, x2p = p, and:
c
      y1p=calphx*y1(i)-cbetax*x1(i)
      y2p=calphx*y2(i)-cbetax*x2(i)
c
c *** Set yl1, the lower limit of integration, and yl2, the
c *** upper limit.
c
      yl1= min(y1p,y2p)
      yl2= max(y1p,y2p)
c
c
c *** Call subroutine INTEGFN to compute scrptF(j,i), j=1,nt.
c
      call INTEGFN(mn,i,ninteg,calphx,cbetax,yl1,yl2,p,scrptF)
c
      return
c
      end
      SUBROUTINE POINT(mn,i,ninteg,scrptF)
c
c
c *** This subroutine computes the coordinates of the image of node i
c *** when i is a point source and calls the integration subroutine
c *** INTEGFP to compute script F for all other nodes on fracture nf.
c
c
      include 'maxdim.blkd/list'
      include 'comfnd.blkd/list'
c
      dimension scrptF(mn,mn)
c
c
      asqdd=radius*radius/(x1(i)*x1(i)+y1(i)*y1(i))
      xii=asqdd*x1(i)
      yii=asqdd*y1(i)
```

```
c
      call INTEGFP(mn,i,ninteg,xii,yii,scrptF)
c
      return
c
      end
      SUBROUTINE RADIAL(mn,i,ninteg,scrptF)
c
c
c *** This subroutine generates the rotation from the local system
c *** of fracture nf to the x'y'system of the radial node i and
c *** calls the integration subroutine INTEGFR to compute scrptF
c *** for all other nodes on fracture nf.
c
c
      include 'maxdim.blkd/list'
      include 'comfnd.blkd/list'
c
      dimension scrptF(mn,mn)
      dimension g(2),x(2),y(2)
c
      x(1)=x1(i)
      x(2)=x2(i)
      y(1)=y1(i)
      y(2)=y2(i)
c
c *** Compute distances between (0,0) and (x1,y1) and between
c *** (0,0) and (x2,y2).
c
      g(1)=sqrt(x(1)*x(1)+y(1)*y(1))
      g(2)=sqrt(x(2)*x(2)+y(2)*y(2))
c
c *** Determine which endpoint is closest to (0,0) and call it k1.
c
      k1=1
      if (g(2) .lt. g(1)) k1=2
      k2=3-k1
c
c *** Use endpoint k2 to determine rotation to x'y' system, thus
c *** the positive x' axis goes from (0,0) to (x(k2),y(k2)).
c
      calphx=x(k2)/g(k2)
      cbetax=y(k2)/g(k2)
c
c *** Determine endpoints of node i in x'y' coordinate system:
c *** xl1 = x'(k1), the lower limit of integration;
c *** xl2 = x'(k2), the upper limit of integration.
c
      xl1=g(k1)
      xl2=g(k2)
c
c *** Adjust sign of xl1.
c
      if ((x(1)*x(2)+y(1)*y(2)) .lt. 0.0) xl1=-xl1
c
c *** Call subroutine INTEGFR to compute scrptF(j,i), j=1,nt.
c
      call INTEGFR(mn,i,ninteg,calphx,cbetax,xl1,xl2,scrptF)
c
      return
```

```
c
      end
c
      SUBROUTINE READ2(ndata)
c
c
c *** This subroutine reads all run identification, flow region
c *** parameters, fracture characteristics, and nodal arrays
c *** written by FMG3D from unit 2.
c
c
      include 'maxdim.blkd/list'
      include 'comfrc.blkd/list'
      include 'comflr.blkd/list'
      include 'comidt.blkd/list'
      include 'comnod.blkd/list'
c
c
c
      read (2,200,end=60) iray,idate
      read (2,210,end=70) (ititle(i),i=1,40)
      read (2,220,end=70) xmesh,ymesh,zmesh
      read (2,230,end=70) rophi,rothe
      read (2,235,end=70)
      read (2,240,end=70) ((frcoef(i,j),j=1,4),i=1,6)
      read (2,250,end=70) nfrac
      read (2,235,end=70)
      read (2,260,end=70) ((frac1(i,j),j=1,8),
     'coefp(i,1),coefp(i,2),coefp(i,3),coefp(i,4),i=1,nfrac)
      read (2,250,end=70) nnode
      read (2,252,end=70) nseg,nsegb
      read (2,235,end=70)
      read (2,290,end=70) ((node1(i,j),j=1,6),i=1,nnode)
      read (2,235,end=70)
      read (2,300,end=70) ((nside(i,j),j=1,2),i=1,nnode)
      read (2,235,end=70)
      read (2,300,end=70) ((node2(i,j),j=1,2),i=1,nnode)
      read (2,305,end=70) node3(1)
      read (2,310,end=70) (node3(i),i=2,node3(1))
      read (2,235,end=70)
      read (2,300,end=70) ((nkey(i,j),j=1,2),i=1,nfrac)
      nnfrac=nfrac
      ndata=1
   50 return
c
c *** Flag normal end of data.
   60 ndata=2
      return
c
c *** Flag irregular end of data.
   70 ndata=3
      return
c
c
  200 format (a7,5x,a9)
  210 format (20a4)
  220 format (3(10x,f10.4))
  230 format (2(10x,f10.4))
  235 format (10x)
  240 format (4f10.4)
```

```
  250 format (10x,i5)
  252 format (2(10x,i5))
  260 format (7f10.4,e12.4,4e12.4)
  290 format (6f10.4)
  300 format (5(3x,2i6))
  305 format (9x,i6,8x)
  310 format (15i5)
c
      end
      SUBROUTINE ROFRAC(nf,rogomu,pkb)
c
c
c *** This subroutine constructs the rotation matrix and trans-
c *** lation coordinates for moving from the global to the
c *** local system of fracture nf.  All nodes on the fracture
c *** are transformed to the local system and numbered from 1 to
c *** nt.  Array ngnn stores the corresponding global node number.
c *** The length, width, and type of each node are also determined.
c *** Fracture transmissivity, permeability * aperture, is
c *** computed in cgs or mks units.
c
c
      include 'maxdim.blkd/list'
      parameter (rlmin=0.005,pmin=0.005,bmin=0.00005)
      parameter (pi180=0.01745329)
c
      include 'comfrc.blkd/list'
      include 'comnod.blkd/list'
      include 'comfnd.blkd/list'
      character*9 typen(3)/'  point  ',' radial  ','nonradial'/
      character*13 bndw/'     boundary'/
c
      dimension rot(3,3)
c
c
c *** Compute fracture transmissivity, permeability * aperture.
c
      pkb=rogomu * frac1(nf,4)**3
c
c *** Form rotation matrix.
c
      phi=frac1(nf,1)*pi180
      theta=frac1(nf,2)*pi180
      rot(1,1)= cos(phi)*cos(theta)
      rot(1,2)=-cos(phi)*sin(theta)
      rot(1,3)=-sin(phi)
      rot(2,1)= sin(theta)
      rot(2,2)= cos(theta)
      rot(2,3)=0.
c     rot(3,1)=coefp(nf,1)
c     rot(3,2)=coefp(nf,2)
c     rot(3,3)=coefp(nf,3)
c
c
c *** Set fracture center for coordinate translation and set radius.
c
      xc=frac1(nf,5)
      yc=frac1(nf,6)
      zc=frac1(nf,7)
      radius=frac1(nf,3)
```

```
c
      write (3,100) nf,radius,frac1(nf,4),pkb
c
c
c
c *** Identification numbers of nodes on fracture nf are stored in
c *** locations n1 to n2 of catalog node3.  Rotate the endpoints
c *** of the nodes and translate them to local fracture system and
c *** store by local numbering system, j=1,nt where nt=n2-n1+1.
c *** Store global node numbers in ngnn(j), j=1,nt.
c
      nt=nkey(nf,1)
      if (nt .gt. maxn) return
      n1=nkey(nf,2)
      n2=n1+nt-1
c
      j=0
      istore=0
      do 50 n=n1,n2
      j=j+1
      nn=node3(n)
      ngnn(j)=nn
c
c *** Transform endpoints of node nn
c
      xcn=node1(nn,1)-xc
      ycn=node1(nn,2)-yc
      zcn=node1(nn,3)-zc
      x1(j)=rot(1,1)*xcn+rot(1,2)*ycn+rot(1,3)*zcn
      y1(j)=rot(2,1)*xcn+rot(2,2)*ycn+rot(2,3)*zcn
c
      xcn=node1(nn,4)-xc
      ycn=node1(nn,5)-yc
      zcn=node1(nn,6)-zc
      x2(j)=rot(1,1)*xcn+rot(1,2)*ycn+rot(1,3)*zcn
      y2(j)=rot(2,1)*xcn+rot(2,2)*ycn+rot(2,3)*zcn
c
c *** Compute the length of node j, rl(j), and determine which type
c *** of node it is, point, radial, or non-radial.
c
   20 rl(j)=sqrt((x2(j)-x1(j))**2 + (y2(j)-y1(j))**2)
      if (rl(j) .gt. rlmin) go to 25
      ntype(j)=1
      go to 30
c
   25 p=x1(j)*y2(j)-x2(j)*y1(j)
      p=abs(p/rl(j))
      ntype(j)=3
      if (p .lt. pmin) ntype(j)=2
c
c *** Compute the width of node nn in the plane of fracture nf and
c *** store one half of the width in bm(j).
c
   30 if (node2(nn,1) .eq. nf) nj=node2(nn,2)
      if (node2(nn,2) .eq. nf) nj=node2(nn,1)
      if (nj .gt. 0) go to 35
c
c *** Node nn is a boundary node.  (Flag for later use by
c *** subroutine storeG).
c
```

```
      bm(j)=-1.0
      istore=1
      write (3,120) j,nn,rl(j),bndw,typen(ntype(j))
      go to 50
c
c *** Node nn is the intersection of fractures nf and nj.
c
   35 cthesq=0.0
      do 40 k=1,3
   40 cthesq=cthesq +coefp(nf,k)*coefp(nj,k)
      cthesq=cthesq*cthesq
      bm(j)=0.5*sqrt(frac1(nj,4)**2/(1.0-cthesq))
      write (3,125) j,nn,rl(j),bm(j),typen(ntype(j))
      if (bm(j) .lt. bmin) bm(j)=-1.0
c
   50 continue
c
c
      return
c
  100 format (1h0,//,18h Nodes on fracture,i5,//,9h   radius,f10.4,
     1 13h     aperture,e13.4,/,17h   transmissivity,e13.4,//,
     2 13h  node number,6x,4hnode,5x,8h1/2 node,6x,4hnode,/,
     3 14h local  global,4x,6hlength,6x,5hwidth,7x,4htype,/)
  120 format (i4,i8,f12.4,a13,3x,a9)
  125 format (i4,i8,f12.4,e13.4,3x,a9)
c
      end
      SUBROUTINE SFLUX(Gstore,mdir,fluxb)
c
c
c *** For each fracture that intersects one or more flow
c *** region boundary planes, this subroutine uses head
c *** values and the stored G matrix to compute flux for
c ***  each node on the fracture.
c
c
      include 'maxdim.blkd/list'
      include 'comnod.blkd/list'
c
      common /aandb/ a(nonz),hd(maxnod,3),fluxn(maxnod,3)
c
      dimension ngnn(maxn),Gstore(1),h(maxn,3),q(maxn,3),
     ,          g(maxn,maxn)
      dimension sumQ(3),fluxb(6,3),fluxt(3)
      character*1 isp(7)/' ','1','2','3','4','5','6'/
      character*4 hdfx(3,2)/3*'head',3*'flux'/
      character*11 direct(3)/'direction 1','direction 2',
     1                       'direction 3'/
c
c
c *** Zero flux arrays.
c
      do m=1,mdir
        fluxt(m)=0.0
        do is=1,6
          fluxb(is,m)=0.0
         enddo
        do nn=1,nnode
          fluxn(nn,m)=0.0
```

```
        enddo
       enddo
c
      write (3,100)
c
c *** For each fracture in turn, set up local node numbering
c *** system and fill h and G matrices.
c
      k=0
   10 k=k+1
      nf=Gstore(k)
      if (nf .gt. maxfrc) go to 50
      write (3,110) nf,(direct(m),m=1,mdir)
      write (3,115) ((hdfx(m,1),hdfx(m,2)),m=1,mdir)
c
      nt=nkey(nf,1)
      n1=nkey(nf,2)
      n2=n1+nt-1
c
      i=0
      do n=n1,n2
        i=i+1
        nn=node3(n)
        ngnn(i)=nn
c
        do m=1,mdir
          h(i,m)=hd(nn,m)
         enddo
c
        do j=i,nt
          k=k+1
          G(j,i)=Gstore(k)
          g(i,j)=g(j,i)
         enddo
       enddo
c
c *** Multiply heads by G matrix to get flux, Q, for each node.
c *** Store Q in fluxn array by global node number and
c *** sum flux on boundary planes in array fluxb.
c
      do i=1,nt
        nn=ngnn(i)
        is=-node2(nn,2)
        isn=1
        do m=1,mdir
          Q(i,m)=0.0
          do j=1,nt
            Q(i,m)=Q(i,m)+G(i,j)*h(j,m)
           enddo
          fluxn(nn,m)=fluxn(nn,m)+Q(i,m)
          if (is.gt.0)then
            fluxb(is,m)=fluxb(is,m)+Q(i,m)
            isn=is+1
           endif
         enddo
        write (3,120) i,nn,isp(isn),((h(i,m),Q(i,m)),m=1,mdir)
       enddo
c
c
c *** Sum flux on fracture nf; the sum should be zero.
```

```
c
      do m=1,mdir
        sumQ(m)=0.0
        do i=1,nt
          sumQ(m)=sumQ(m)+Q(i,m)
         enddo
       enddo
c
      write (3,130) (sumQ(m),m=1,mdir)
c
      go to 10
c
   50 write (3,140) (direct(m),m=1,mdir)
      write (3,145) ((hdfx(m,1),hdfx(m,2)),m=1,mdir)
      do 51 nn=1,nnode
      is=-node2(nn,2)
      if (is .le. 0) is=0
   51 write (3,150) nn,isp(is+1),((hd(nn,m),fluxn(nn,m)),
     1                m=1,mdir)
      write (3,160) (direct(m),m=1,mdir)
      do 54 is=1,6
      do 52 im=1,mdir
   52 fluxt(im)=fluxt(im)+fluxb(is,im)
   54 write (3,170) is,(fluxb(is,m),m=1,mdir)
      write (3,180) (fluxt(m),m=1,mdir)
c
c
   90 return
c
c
c
  100 format (1h1,22h *** FLUX AND HEAD ***,//,
     1 49h FLUX THROUGH EACH FRACTURE AND HEAD AT EACH NODE)
  110 format (1h0,9h Fracture,i5,//,20h  node number   side,
     1        7x,a11,2(13x,a11))
  115 format (1h ,24h local  global   no.    ,6(a4,8x))
  120 format (i4,i8,6x,a1,3(f10.4,e14.4))
  130 format (1h0,14h Sum of fluxes,5x,3e24.4)
  140 format (1h0,45h TOTAL FLUX THROUGH EACH NODE AND HEAD VALUES,//,
     1       12h node   side,7x,a11,2(13x,a11))
  145 format (1h ,16h  no.   no.     ,6(a4,8x))
  150 format (i4,6x,a1,3(f10.4,e14.4))
  160 format (1h0,39h TOTAL FLUX THROUGH EACH BOUNDARY PLANE,//,
     1        5h side,4x,3(a11,3x))
  170 format (i4,2x,3e14.4)
  180 format (6h0total,3e14.4)
c
      end
      SUBROUTINE STOREG(maxfrc,maxn,nf,nt,G,Gstore,k)
c
c
c *** This subroutine stores G matrices for fractures that
c *** intersect one or more flow region boundary planes.
c
c
      dimension G(maxn,maxn),Gstore(1)
c
c
      k=k+1
      Gstore(k)=nf
```

```
c
      do 20 j=1,nt
      do 20 i=j,nt
c
      k=k+1
      Gstore(k)=G(i,j)
   20 continue
c
c *** Flag last entry for subroutine SFLUX.
      Gstore(k+1) = maxfrc + 10
c
      return
c
      end
```

APPENDIX E

Program DIMES Organization and Arrays

Table E-1. Description of DIMES Program Variables

Variable	Description	How Value is Assigned
answer	Answer to questions on the screen	Read from keyboard in DIMES
delta	Angle measured from center of fracture between the two endpoints of a segment on the fracture	Set in DIMES
dgene	Scaling factor for the small orientation box	Computed in INIPLOT
dmax	Scaling factor for the main plot; maximum distance between a point on a disc and the center of the plot	Computed in INIPLOT
dplan	Perpendicular distance from origin to plane of a fracture	Computed in DIMES
idot	Counter for the number of dotted line segments already processed	Set and updated in DIMES and DRAW
ilast	Number of the last segment processed	Set and updated in DIMES
iopen	Code for status of file DIMES.INP = 0 file is closed = 1 file is opened	Set in RDATA and DIMES
iplotc	Code for next plot: = 0 change cubes in subregion = 1 keep same subregion	Set in DIMES from value of answer
iplotv	Code for next plot = 0 change viewphi,viewthe and npol = 1 keep same viewphi,viewthe and npol	Set in DIMES from value of answer
iseg	Number of the segment currently processed	Set and used in many parts of program
ivoid	Number of the last entry in the segments arrays which has been voided (segment completely hidden)	Set and updated in DIMES
maxcub	Maximum number of cubes maxcub = maxord**3, used for dimensioning arrays	Set in parameter statement in DIMES
maxfr	Maximum number of discs, used	Set in parameter

	for dimensioning arrays	statement in DIMES
maxord	Maximum value for the cube division parameter iord, used for dimensioning arrays	Set in parameter statement in DIMES
maxseg	Maximum number of plot segments, used for dimensioning arrays	Set in parameter statement in DIMES
ndot	Number of segments to be plotted in dotted lines	Computed in DIMES
nflow	Number of current flow region	Interactive mode: set and updated in DIMES from value of answer Batch mode: updated in RDATA from value of chap
nfrac	Number of fractures, then number of discs	Read in RDATA, then updated in RDATA
nseg	Size of the useful part of of the segment arrays (number of segments may be smaller due to voids in arrays)	Set in rdata, updated in DIMES, INIPLOT
rad	Radius of current fracture ifr rad = frac(3,ifr)	Set in DIMES
t1,t2	Roots of the equation for the intersection of segment iseg and current ellipse, parametered on the line supporting segment iseg, t1 < t2 t1 or t2 = 0: intersection on first endpoint of segment iseg t1 or t2 = 1: intersection on second endpoint of segment iseg	Computed in DIMES
viewang	Intermediate angle value for calculation of array viewmat	Set in DIMES
xaper	x coordinate of vector normal to the current fracture and with length equal to the rescaled aperture.	Computed in DIMES
xnorm	x coordinate of normal to current fracture	Computed in DIMES

x1,x2	x coordinates of endpoints of current segment x1 = end(1,iseg) x2 = end(2,iseg)	Set in DIMES
yaper	y coordinate of vector normal to the current fracture and with length equal to the rescaled aperture.	Computed in DIMES
ynorm	z coordinate of normal to current fracture	Computed in DIMES
y1,y2	y coordinates of endpoints of current segment y1 = end(3,iseg) y2 = end(4,iseg)	Set in DIMES
zaper	z coordinate of vector normal to the current fracture and with length equal to the rescaled aperture.	Computed in DIMES
znorm	z coordinate for normal to current fracture	Set in DIMES
z1,z2	z coordinates of endpoints of current segment z1 = end(5,iseg) z2 = end)6,iseg)	

Table E-2. Description of DIMES Program Arrays

Variable	Description
index(maxfrc)	Reference index for fracture numbers used in DIMES instead of permitting fractures when they are sorted, ifr = 1,nfrac index(ifr) = entry number in fracture arrays of disc in position ifr after the sorting
inew(maxfrc)	Reference index for fracture numbers when the fractures are reordered in sequence in RDATA. Used to reset the array ifint,ifr=1,nfrac inew(ifr) = new number of fracture numbered ifr in the input file.
inext(0:maxseg-1)	Key to arrays end and ifint: number of next entry in these arrays. Set and updated in DIMES. iseg = 0,nseg inext(iseg) = if entry iseg contains a segment, number of next segment if entry iseg is void (hidden segment), number of next void entry.
lplot(maxcub)	Selection code for cubes, set up and used in subroutine RDATA, ic=1,ncub lplot(ic) = 1 fractures in cube ic will be plotted lplot(ic) = 0 fractures in cube ic will not be plotted
matrix(3,3)	Local name of a rotation matrix when used or computed by subroutine ROTATE, or ROTFRAC or computed by ROTMAT.
romat(3,3)	Flow region rotation matrix, computed in ROTMAT and used in INIPLOT
trig(4,maxfrc)	sines and cosines of the fracture orientation angles ϕ and θ, set in subroutine RDATA, and in DIMES, updated in subroutine ROTFRAC ifr = 1,nfrac trig(1,ifr) = cos(ϕ) trig(2,ifr) = sin(ϕ) trig(3,ifr) = cos(θ) trig(4,ifr) = sin(θ)
viewmat(3,3)	Viewpoint rotation matrix, computed in ROTMAT
xbox(24,2)	x coordinates of the endpoints of the segments making up the "orientation box" plot, i = 1,24 if i $\leq$ 12, segment in generation region if i $>$ 12, segment on flow region.

xyz(3)	Coordinates array used each time a point or vector needs to be rotated. Used in most subroutines.
ybox(24,2)	y coordinates of the endpoints of the segments making up the "orientation box" plot.
zbox(24,2)r	z coordinates of the endpoints of the segments making up the "orientation box" plot.

Table E-3. DIMES Subroutine Outline

DIMES	RDATA		
	ROTMAT		
	INIPLOT	BOXES	ROTATE
		ROTMAT	
		ROTATE	
		BOXES	ROTATE
	ROTFRAC	ROTATE	
	ROTATE		
	DRAW	[DISPLA System	Subroutines]

Table E-4. Description of DIMES Subroutines

Subroutine	Description
BOXES	Rotates, scales and translates the small plot of generation region or flow region, depending on parameter k (k=0 generation region -k=1 flow region. given: end(6,nseg),xgene,xmesh,ygene,ymesh, zgene,zmesh,viewmat(3,3),dgene,dmax returns: xbox(i,2),ybox(i,2),zbox(i,2) i = 12k+1, 12k+12
DRAW	Draws the plot either on screen or in a plot file. given: ititle1(20),ititle2(20),dmax,ndot,end(6,nseg), inext(0:nseg),xbox(24,2),ybox(24,2),zbox(24,2) draws a plot uses the DISPLA software
INIPLOT	Initializes plot parameters: scaling, orientation box. given: xmesh,ymesh,zmesh,xgene,ygene,zgene,radmax, apmax,frac(7,nfrac),rophi,rothe returns: dmax,dgene,xbox(24,2),ybox(24,2),zbox(24,2), modified frac(7,nfrac),end(6,nseg) and nseg uses subroutines: BOXES ROTATE ROTMAT
RDATA	Reads and rearranges the data from input files and/or keyboard. reads: ititle1(20),ititle2(20),xgene,ygene,zgene, xmesh,ymesh,zmesh,rophi,rothe,nfrac,nint, nout,iord,apmax,radmax,numfrc(ncub),numinc(ncub), ifrok(ncub+1),viewphi,viewthe,npol,ix1,ix2,iy1, iy2,iz1,iz2,frac(7,nfrac),end(6,nint+nout), ifint(2,nint+nout) returns: ncub,ndot,trig(4,nfrac), modified nfrac,frac, end,ifint
ROTATE	Rotates a vector or point according to a specified matrix. given: xyz(3),matrix(3,3) returns: modified xyz
ROTFRAC	Rotates a fracture ifrac: computes the new sines and cosines of the fracture orientation angles after rotation according to a specified matrix. given: ifrac,matrix(3,3),trig(4,ifrac) returns: modified trig
ROTMAT	Computes a rotation matrix for specified rotation angles. given: phi, theta

APPENDIX F

Listing of Program DIMES

include file for DIMES:

```
file COMMON.DIM
     parameter (maxfr=1000,maxseg=70000,maxord=6,
    ,           maxcub=maxord**3)
     common/boxes/xbox(24,2),ybox(24,2),zbox(24,2),
    ,             xgene,ygene,zgene,dgene
     common/dimen/nfrac,nint,nseg
     common/fract/frac(9,maxfr+1)
     dimension trig(9,maxfr)
     equivalence (frac(8,1),trig(1,1))
     common/ngraph/nflow,iopen,iplotc,iplotv,lplot(maxcub)
     real*4 end(9,maxseg)
     integer*4 ifint(9,maxseg),inext(9,0:maxseg-1)
     common/plot/end,ndot
     equivalence (end(1,1),ifint(1,1),inext(1,0))
     common/roflo/rophi,rothe
     common/roview/viewphi,viewthe,viewmat(3,3),npol
     common/size/xmesh,ymesh,zmesh,apmax,radmax,dmax
     integer*4 ititle1(20),ititle2(20)
     common/titl/ititle1,ititle2
```

```
c
      program dimes
c
c     plotting program for fmg3d. written feb-may 1986 by db
c
c
c     input data is in files dimesnn.dat and is:
c
c              * centers, radii, and orientation angles
c                of all fractures;
c              * endpoints of all intersections.
c
c     program control is given at keyboard (interactive runs) or in
c     file dimes.inp (batch runs), and is:
c
c               * view angles, number of segments defining each fracture
c                 disc;
c               * selection of bloc subregions to be displayed.
c               * output device
c
      include 'common.dim'
      dimension xyz(3)
      integer index(maxfr),answer
      nflow=1
      iopen=0
      iplotc=0
      iplotv=0
c
c     read data from input files and screen .
c
   10 call rdata
c
      viewang=90.-viewphi
      delta=6.28318/float(npol)
c
c     compute viewpoint rotation matrix
c
      call rotmat(viewang,viewthe,viewmat)
c
c     initialise plot parameters
c
      call iniplot
c
c     loop over intersections: rotate endpoints
c     according to viewpoint.
c
      do iseg=1,nseg
        do j=1,2
          xyz(1)=end(j,iseg)
          xyz(2)=end(j+2,iseg)
          xyz(3)=end(j+4,iseg)
          call rotate(xyz,viewmat)
          end(j,iseg)=xyz(1)
          end(j+2,iseg)=xyz(2)
          end(j+4,iseg)=xyz(3)
          ifint(j+6,iseg)=1+(ifint(j+6,iseg)-1)*2
        enddo
       enddo
c
c     first loop over the fractures: rotating them
```

```
c     according to viewpoint and dividing them
c     into segments.
c
      do ifr=1,nfrac-1,2
c
c       compute the sines and cosines of the normal to fracture
c       # ifrac after rotation by [matrix].
c
        call rotfrac(ifr,viewmat)
c
c       compute new center coordinates
c
        xyz(1)=frac(5,ifr)
        xyz(2)=frac(6,ifr)
        xyz(3)=frac(7,ifr)
        call rotate(xyz,viewmat)
        frac(5,ifr)=xyz(1)
        frac(6,ifr)=xyz(2)
        frac(7,ifr)=xyz(3)
c
c       divide fractures in two discs to give perspective
c
        xaper=trig(2,ifr)*trig(3,ifr)*frac(4,ifr)
        yaper=-trig(2,ifr)*trig(4,ifr)*frac(4,ifr)
        zaper=trig(1,ifr)*frac(4,ifr)
c
        if(zaper.lt.0.)then
          xaper=-xaper
          yaper=-yaper
          zaper=-zaper
         endif
c
        ifr1=ifr+1
        frac(5,ifr1)=frac(5,ifr)-xaper
        frac(6,ifr1)=frac(6,ifr)-yaper
        frac(7,ifr1)=frac(7,ifr)-zaper
        frac(3,ifr1)=frac(3,ifr)
        trig(1,ifr1)=trig(1,ifr)
        trig(2,ifr1)=trig(2,ifr)
        trig(3,ifr1)=trig(3,ifr)
        trig(4,ifr1)=trig(4,ifr)
        do iplus=0,1
c
c         divide circle into segments.
c
          rad=frac(3,ifr)
          ifplus=ifr+iplus
          do ipol=0,npol-1
            nseg=nseg+1
            ifint(7,nseg)=ifplus
            ifint(8,nseg)=0
            a=rad*cos(delta*ipol)
            b=rad*sin(delta*ipol)
            c=a*trig(1,ifr)
            end(1,nseg)=frac(5,ifr)+c*trig(3,ifr)
     ,                  +b*trig(4,ifr)-iplus*xaper
            end(3,nseg)=frac(6,ifr)-c*trig(4,ifr)
     ,                  +b*trig(3,ifr)-iplus*yaper
            end(5,nseg)=frac(7,ifr)-a*trig(2,ifr)-iplus*zaper
            if (ipol.gt.0)then
```

```
              end(2,nseg-1)=end(1,nseg)
              end(4,nseg-1)=end(3,nseg)
              end(6,nseg-1)=end(5,nseg)
             endif
            enddo
           end(2,nseg)=end(1,nseg-npol+1)
           end(4,nseg)=end(3,nseg-npol+1)
           end(6,nseg)=end(5,nseg-npol+1)
c
c          initialize index, and put into the aperture array
c          frac(4,ifr) the parameter used for ordering the fractures
c          (the aperture is not used any more).
c
           index(ifplus)=ifplus
           frac(4,ifplus)=(frac(7,ifplus)+dmax)*abs(trig(1,ifplus))
     ,                    *frac(3,ifplus)*frac(3,ifplus)

         enddo
        enddo
c
c     second loop over the fractures: ordering index
c     according to frac(4,ifr).
c
      j1=2
      i2=nfrac
      do while (i2.ge.j1)
        j2=i2
        i0=1
        i1=0
        i2=0
        do 20 j=j1,j2
          inj =index(j)
          inj1=index(j-1)
          if (frac(4,inj1).ge.frac(4,inj)) go to 20
          index(j)=inj1
          index(j-1)=inj
          i2=j-1
          i1=i1+i0*i2
          i0=0
   20     continue
        j1=max0(i1,2)
       enddo
c
c     initialise inext
c
      do iseg=0,maxseg-1
        inext(9,iseg)=iseg+1
       enddo
c
c     third loop over the fractures:
c     hiding
c
      ivoid=-1
      inext(9,nseg)=-1
      do ifr=1,nfrac
        ifrac=index(ifr)
        if(trig(1,ifrac).le.1e-5)goto 40
        rad=frac(3,ifrac)
        xnorm=trig(2,ifrac)*trig(3,ifrac)
        ynorm=-trig(2,ifrac)*trig(4,ifrac)
```

```
      znorm=trig(1,ifrac)
      dplan=-xnorm*frac(5,ifrac)-ynorm*frac(6,ifrac)
     ,       -znorm*frac(7,ifrac)
      idot=1
      iseg=inext(9,0)
      ilast=0
c
c     loop over the segments not already hiden.
c
      do while (iseg.gt.0)
c
        x1=end(1,iseg)
        x2=end(2,iseg)
        y1=end(3,iseg)
        y2=end(4,iseg)
        z1=end(5,iseg)
        z2=end(6,iseg)
c
c       is segment part of fracture # ifrac or of intersection
c       on fracture # ifrac?
c
        if(ifint(7,iseg).eq.ifrac.or.
     ,     ifint(8,iseg).eq.ifrac)goto 30
c
c       is segment close to ellipse ?
c
        xlen=x2-x1
        ylen=y2-y1
        slen=sqrt(xlen*xlen+ylen*ylen)
        dismin=rad+slen/2.
        xdist=(x1+x2)/2.-frac(5,ifrac)
        ydist=(y1+y2)/2.-frac(6,ifrac)
        d2=xdist*xdist+ydist*ydist
        if(d2.ge.dismin*dismin)goto 30
c
c       divide segments intersecting fracture plane
c
        zplan1=-(dplan+xnorm*x1+ynorm*y1)/znorm
        zplan2=-(dplan+xnorm*x2+ynorm*y2)/znorm
c
c
        dz1=z1-zplan1
        dz2=z2-zplan2
        if(dz1.ge.0..and.dz2.ge.0.)goto 30
        if(dz1*dz2.lt.0.)then
          ddz=dz2-dz1
          if(dz1.ge.0.)then
            x1=(x1*dz2-x2*dz1)/ddz
            y1=(y1*dz2-y2*dz1)/ddz
            z1=(z1*dz2-z2*dz1)/ddz
            end(2,iseg)=x1
            end(4,iseg)=y1
            end(6,iseg)=z1
           else
            x2=(x1*dz2-x2*dz1)/ddz
            y2=(y1*dz2-y2*dz1)/ddz
            z2=(z1*dz2-z2*dz1)/ddz
            end(1,iseg)=x2
            end(3,iseg)=y2
            end(5,iseg)=z2
```

```
            endif
           if(ivoid.lt.0)then
             nseg=nseg+1
             inext(9,nseg)=ivoid
             ivoid=nseg
            endif
           if(idot.le.ndot)ndot=ndot+1
           idot=idot+1
           next=inext(9,iseg)
           inext(9,iseg)=ivoid
           ilast=iseg
           iseg=ivoid
           ivoid=inext(9,iseg)
           inext(9,iseg)=next
           do i=7,8
            ifint(i,iseg)=ifint(i,ilast)
           enddo
          endif
c
c         segment under fracture plane and close to ellipse.
c         compute segment-ellipse intersection(s).
c
          xcen=frac(5,ifrac)*trig(3,ifrac)
     ,         -frac(6,ifrac)*trig(4,ifrac)
          ycen=frac(5,ifrac)*trig(4,ifrac)
     ,         +frac(6,ifrac)*trig(3,ifrac)
          xp1=x1*trig(3,ifrac)-y1*trig(4,ifrac)-xcen
          xp1=xp1/trig(1,ifrac)
          yp1=x1*trig(4,ifrac)+y1*trig(3,ifrac)-ycen
          xp2=x2*trig(3,ifrac)-y2*trig(4,ifrac)-xcen
          xp2=xp2/trig(1,ifrac)
          yp2=x2*trig(4,ifrac)+y2*trig(3,ifrac)-ycen
c
          a=(xp2-xp1)*(xp2-xp1)+(yp2-yp1)*(yp2-yp1)
          b=xp1*(xp2-xp1)+yp1*(yp2-yp1)
          b=2.*b
          c=xp1*xp1+yp1*yp1-rad*rad
c
          delt=b*b-4.*a*c
          if(delt.le.0.)goto 30
c
          root=sqrt(delt)
          t1=-(b+root)/(2*a)
          if(t1.ge.1.)goto 30
          t2=(-b+root)/(2*a)
          if(t2.le.0.)goto 30
c
          if(t1.le.0.)then
            if(t2.ge.1.)then
c
c             segment completely hidden
c
              inext(9,ilast)=inext(9,iseg)
              if(iseg.eq.nseg)then
                nseg=nseg-1
               else
                inext(9,iseg)=ivoid
                ivoid=iseg
               endif
              iseg=ilast
```

```
            if(idot.le.ndot)ndot=ndot-1
            idot=idot-1
           else
c
c            first endpoint hidden
c
            x1=x1+(x2-x1)*t2
            y1=y1+(y2-y1)*t2
            z1=z1+(z2-z1)*t2
           endif
          else
           if(t2.ge.1.)then
c
c            second endpoint hidden
c
            x2=x1+(x2-x1)*t1
            y2=y1+(y2-y1)*t1
            z2=z1+(z2-z1)*t1
           else
c
c            middle of segment hidden
c
            end(1,iseg)=x1
            end(2,iseg)=x1+(x2-x1)*t1
            end(3,iseg)=y1
            end(4,iseg)=y1+(y2-y1)*t1
            end(5,iseg)=z1
            end(6,iseg)=z1+(z2-z1)*t1
            x1=x1+(x2-x1)*t2
            y1=y1+(y2-y1)*t2
            z1=z1+(z2-z1)*t2
            if(ivoid.lt.0)then
              nseg=nseg+1
              inext(9,nseg)=ivoid
              ivoid=nseg
             endif
            next=inext(9,iseg)
            inext(9,iseg)=ivoid
            ilast=iseg
            iseg=ivoid
            ivoid=inext(9,iseg)
            inext(9,iseg)=next
            if(idot.le.ndot)ndot=ndot+1
            idot=idot+1
            do i=7,8
             ifint(i,iseg)=ifint(i,ilast)
            enddo
           endif
          endif
c
c         store new coordinates of segment if modified
c
   30     if(iseg.ne.ilast)then
           end(1,iseg)=x1
           end(2,iseg)=x2
           end(3,iseg)=y1
           end(4,iseg)=y2
           end(5,iseg)=z1
           end(6,iseg)=z2
          endif
```

```
          ilast=iseg
          iseg=inext(9,iseg)
          idot=idot+1
         enddo
   40   continue
       enddo
c
c     draw plot
c
   50 call draw
c
c     ask for new plot?
c
      write(6,6001)
 6001 format(' do you want to change the subregion to plot?',
     '       ' (y/n)',$)
      read(5,5001,end=10)answer
      if(answer.eq.'y'.or.answer.eq.'Y')then
        iplotc=0
       else
        iplotc=1
       endif
      write(6,6002)
 6002 format(' do you want to change the viewpoint? (y/n)',$)
      read(5,5001)answer
 5001 format(a1)
      if(answer.eq.'y'.or.answer.eq.'Y')then
        iplotv=0
       else
        iplotv=1
       endif
      write(6,6003)
 6003 format(' do you want to change the flow region '/
     '       ' (stop if last flow region) ? (y/n)    ',$)
      read(5,5001,end=10)answer
      if(answer.eq.'y'.or.answer.eq.'Y')then
        nflow=nflow+1
       else
        if(iplotc.eq.1.and.iplotv.eq.1)goto 50
       endif
      goto 10
      end
c
c
c
      subroutine boxes(k)
c
c     compute small plot of generation region and flow region
c
      include 'common.dim'
      dimension xyz(3)
c
      kk=12*k
      do i=1,12
        do j=1,2
          xyz(1)=end(j,nseg+i)*(1.+(xgene/xmesh-1.)*float(1-k))
          xyz(2)=end(j+2,nseg+i)*(1.+(ygene/ymesh-1.)*float(1-k))
          xyz(3)=end(j+4,nseg+i)*(1.+(zgene/zmesh-1.)*float(1-k))
          call rotate(xyz,viewmat)
          xbox(kk+i,j)=(xyz(1)/dgene+4.5)*dmax/3.5
```

```
          ybox(kk+i,j)=(xyz(2)/dgene-2.5)*dmax/3.5
        enddo
      enddo
      return
      end
c
c
      subroutine draw
c
c     draws the plot
c
      include 'common.dim'
      dimension xaray(2),yaray(2)
c
      call nomin8
      call nobrdr
      call area2d(7.,10.)
      call thkfrm(0.03)
      call frame
      call messag(ititle1,60,0.3,9.5)
      call messag(ititle2,60,0.3,9.)
      xorig=-dmax
      yorig=-dmax*5.5/3.5
      xstp=dmax/10.
      ystp=dmax/10.
      xmax=dmax
      ymax=dmax*4.5/3.5
      call graf(xorig,xstp,xmax,yorig,ystp,ymax)
      iseg=inext(9,0)
      if(ndot.ne.0)then
        idot=1
        call dot
      endif
      do while (iseg.gt.0)
c
c       rotate to have correct plot on screen
c
        xaray(1)=end(3,iseg)
        xaray(2)=end(4,iseg)
        yaray(1)=-end(1,iseg)
        yaray(2)=-end(2,iseg)
        call curve(xaray,yaray,2,0)
        if(idot.eq.ndot)call reset('dot')
        iseg=inext(9,iseg)
        idot=idot+1
      enddo
c
c
c     draw frame around small boxes
c
      call thkcrv(0.03)
      xaray(1)=-dmax
      xaray(2)=-dmax*1.5/3.5
      yaray(1)=-dmax
      yaray(2)=-dmax
      call curve(xaray,yaray,2,0)
      xaray(1)=xaray(2)
      yaray(1)=-dmax*5.5/3.5
      call curve(xaray,yaray,2,0)
c
```

```
c     draw boxes
c
      call thkcrv(0.02)
      do ibox=1,12
        xaray(1)=ybox(ibox,1)
        xaray(2)=ybox(ibox,2)
        yaray(1)=-xbox(ibox,1)
        yaray(2)=-xbox(ibox,2)
        call curve(xaray,yaray,2,0)
       enddo
      call reset('thkcrv')
      do ibox=13,24
        xaray(1)=ybox(ibox,1)
        xaray(2)=ybox(ibox,2)
        yaray(1)=-xbox(ibox,1)
        yaray(2)=-xbox(ibox,2)
        call curve(xaray,yaray,2,0)
       enddo
      call endpl(0)
      return
      end
c
c
c
      subroutine iniplot
c
c     initialise plot parameters
c
      include 'common.dim'
      dimension romat(3,3),romat1(3,3),xyz(3)
c
      dmax=xmesh*xmesh+ymesh*ymesh+zmesh*zmesh
      dmax=sqrt(dmax/4.)+radmax
c
      dgene=xgene*xgene+ygene*ygene+zgene*zgene
      dgene=sqrt(dgene/4.)+radmax
c     dgene=3.5*dgene/dmax
c
      thicmin=0.003*dmax
      thicmax=0.008*dmax
      do ifrac=1,nfrac-1,2
        frac(4,ifrac)=thicmin+(thicmax-thicmin)
     *                 *frac(4,ifrac)/apmax
       enddo
c
c     compute box endpoints
c
      x=xmesh/2.
      y=ymesh/2.
      z=zmesh/2.
c
      end(1,nseg+1)=x
      end(3,nseg+1)=-y
      end(5,nseg+1)=-z
      end(1,nseg+2)=x
      end(3,nseg+2)=y
      end(5,nseg+2)=-z
      end(1,nseg+3)=x
      end(3,nseg+3)=y
      end(5,nseg+3)=z
```

```
      end(1,nseg+4)=x
      end(3,nseg+4)=-y
      end(5,nseg+4)=z
      end(1,nseg+5)=x
      end(3,nseg+5)=-y
      end(5,nseg+5)=-z
      end(1,nseg+6)=-x
      end(3,nseg+6)=-y
      end(5,nseg+6)=-z
      end(1,nseg+7)=-x
      end(3,nseg+7)=y
      end(5,nseg+7)=-z
      end(1,nseg+8)=-x
      end(3,nseg+8)=y
      end(5,nseg+8)=z
      end(1,nseg+9)=-x
      end(3,nseg+9)=-y
      end(5,nseg+9)=z
      end(1,nseg+10)=-x
      end(3,nseg+10)=-y
      end(5,nseg+10)=-z
c
      do iseg=nseg+1,nseg+9
        end(2,iseg)=end(1,iseg+1)
        end(4,iseg)=end(3,iseg+1)
        end(6,iseg)=end(5,iseg+1)
       enddo
c
      end(1,nseg+10)=x
      end(2,nseg+10)=-x
      end(3,nseg+10)=y
      end(4,nseg+10)=y
      end(5,nseg+10)=-z
      end(6,nseg+10)=-z
      end(1,nseg+11)=x
      end(2,nseg+11)=-x
      end(3,nseg+11)=y
      end(4,nseg+11)=y
      end(5,nseg+11)=z
      end(6,nseg+11)=z
      end(1,nseg+12)=x
      end(2,nseg+12)=-x
      end(3,nseg+12)=-y
      end(4,nseg+12)=-y
      end(5,nseg+12)=z
      end(6,nseg+12)=z
c
c     generation region
c
      end(1,nseg+5)=end(1,nseg+5)+radmax*xmesh/xgene
      end(4,nseg+6)=end(4,nseg+6)+radmax*ymesh/ygene
      end(5,nseg+9)=end(5,nseg+9)+radmax*zmesh/zgene
      call boxes(0)
      end(1,nseg+5)=end(1,nseg+5)+radmax*(1.-xmesh/xgene)
      end(4,nseg+6)=end(4,nseg+6)+radmax*(1.-ymesh/ygene)
      end(5,nseg+9)=end(5,nseg+9)+radmax*(1.-zmesh/zgene)
c
c
c     rotate box according to rophi and rothe
c
```

```
c     compute flowregion rotation matrix
c
      the=-rothe
      call rotmat(rophi,the,romat1)
      do i=1,3
        do j=1,3
          romat(i,j)=romat1(j,i)
         enddo
       enddo
c
c     loop over box segments: rotate endpoints
c     according to romat.
c
      do iseg=nseg+1,nseg+12
        do j=1,2
          xyz(1)=end(j,iseg)
          xyz(2)=end(j+2,iseg)
          xyz(3)=end(j+4,iseg)
          call rotate(xyz,romat)
          end(j,iseg)=xyz(1)
          end(j+2,iseg)=xyz(2)
          end(j+4,iseg)=xyz(3)
          ifint(j+6,iseg)=0
         enddo
       enddo
c
c     flow region
c
      call boxes(1)
c
      nseg=nseg+12
      return
      end
c
c
c
      SUBROUTINE NOMIN8
C
 10   WRITE(6,20)
 20   FORMAT(' ENTER DEVICE TYPE INDEX # (1 THRU 7).....',/,
     1  10X,' 1 = tek4010            5 = imagen      '/,
     2  10X,' 2 = tek4014            6 = talaris     '/,
     2  10X,' 3 = tek4105            7 = zeta        '/,
     3  10X,' 4 = Peters hp plotter                  '/)
C
      READ(5,30,end=25)MYDEV
 30   FORMAT(I2)
      IF(MYDEV.LE.0.OR.MYDEV.GT.7) GO TO 900
      IF(MYDEV.EQ. 1) call tekall(4010,960,2,0,0)
      IF(MYDEV.EQ. 2) call tekall(4014,960,2,0,0)
      IF(MYDEV.EQ. 3) call tk41(4105)
      IF(MYDEV.EQ. 4) CALL H7221T
      IF(MYDEV.EQ. 5) CALL IMAGEN
      IF(MYDEV.EQ. 6) CALL TALARS
      IF(MYDEV.EQ. 7) CALL PZETA
      RETURN
 25   call talars
      return
C
 900  WRITE(6,898)
```

```
 898  FORMAT(' YOUR CHOICE IS NOT VALID')
      GO TO 10
C
      END
c
c
c
      subroutine rdata
      include 'common.dim'
      character*4 chap
      character*11 file
      character*10 form
      parameter (pi180=0.0174532925)
      dimension numfrc(maxcub),numinc(maxcub),ifrok(maxcub+1),
     '          inew(maxfr)
c
c
c
   10 write(file,20)nflow
   20 format('dimes',i2.2,'.dat')
      open(unit=1,file=file,status='old',readonly,err=60)
      read(1,1001)ititle1,ititle2
      read(1,1002)xgene,ygene,zgene
      read(1,1002)xmesh,ymesh,zmesh
      read(1,1002)rophi,rothe
      read(1,1003)nfrac,nint,nout
      read(1,1003)iord
      read(1,1002)apmax,radmax
      iord2=iord**2
      ncub=iord**3
      read(1,1004)(numfrc(ic),ic=1,ncub)
      read(1,1004)(numinc(ic),ic=1,ncub)
      read(1,1004)(ifrok(ic),ic=1,ncub+1)
c
c     read view angles and discretization from keyboard
c
      if(iplotv.eq.0)then
        write(6,6001)
 6001   format(' Enter view angles:'//'   phi......',$)
        read(5,*,end=30)viewphi
        write(6,6002)
 6002   format('   theta....',$)
        read(5,*)viewthe
        write(6,6003)
 6003   format(/' Number of segments per disc...',$)
        read(5,*)npol
       endif
c
c     read on which cubes to draw the fractures
c
      if(iplotc.eq.0)then
        write(6,6100)iord,iord,iord,iord
 6100   format(//' The generation region has been subdivided in '/
     '          ' ',2(i2,' by '),i2,' cubes.'//
     '          ' The subregion to be plotted is specified by the '/
     '          ' first and last cube numbers in each direction'/
     '          ' (between 1 and ',i2,' )'/
     '          ' Enter ix1,ix2,iy1,iy2,iz1,iz2 (free format)')
        read(5,*)ix1,ix2,iy1,iy2,iz1,iz2
       endif
```

```
      goto 50
c
c     read from input file if batch job
c
   30 continue
      if(iopen.eq.0)then
        open(unit=2,file='dimes.inp',status='old',readonly)
        iopen=1
       endif
      read(2,1005,end=40)chap
      if(chap.eq.'flow'.or.chap.eq.'FLOW')goto 45
      if(chap.eq.'view'.or.chap.eq.'VIEW')then
        read(2,1006,end=40)viewphi,viewthe,npol
        read(2,1005)chap
       endif
      read(2,1007)ix1,ix2,iy1,iy2,iz1,iz2
      goto 50
   40 close(unit=2)
      iopen=0
   45 nflow=nflow+1
      goto 10
   50 if(iplotc.eq.0)then
c
c       reinitialize
c
        do i=1,ncub
          lplot(i)=0
         enddo
c
        do ix=ix1-1,ix2-1
          do iy=iy1-1,iy2-1
            do iz=iz1-1,iz2-1
              ic=ix+iy*iord+iz*iord2+1
              lplot(ic)=1
             enddo
           enddo
          enddo
       endif
c
c     read characteristics of fractures and intersections
c     for each cube
c
c     initialize
c
      do ifr=1,nfrac
        frac(3,ifr)=0.
       enddo
      do ient=1,nout+nint
        ifint(7,ient)=0
       enddo
c
      n=nout
      do ic=1,ncub
        if(lplot(ic).eq.1)then
c
c         read in selected cubes
c
          read(1,1004)icub
          if(numfrc(ic).ne.0)then
            do ifrac=1,numfrc(ic)
```

```
            read(1,1008)i,(frac(k,i),k=1,7)
           enddo
          if(ifrok(ic).ne.ifrok(ic+1))then
           do iout=ifrok(ic),ifrok(ic+1)-1
             read(1,1009)i,end(1,i),end(3,i),end(5,i),
     ,                     end(2,i),end(4,i),end(6,i),
     ,                     ifint(7,i),ifint(8,i)
            enddo
           endif
          do iint=1,numinc(ic)
            read(1,1009)i,end(1,i+n),end(3,i+n),end(5,i+n),
     ,                   end(2,i+n),end(4,i+n),end(6,i+n),
     ,                   ifint(7,i+n),ifint(8,i+n)
           enddo
         endif
        else
c
c         skip non selected cubes
c
         nlin=1+numfrc(ic)+ifrok(ic+1)-ifrok(ic)+numinc(ic)
         if(nlin.eq.1)then
          read(1,1004)icub
         else
          nlin1=nlin-1
          write(form,2000)nlin1
 2000     format('(',i5,'(/))')
          read(1,form)
         endif
        endif
       enddo
c
c     renumber fractures in sequence
c
      i=0
      c=1.
      do ifr=1,nfrac
        if(frac(3,ifr).gt.0.)then
          i=i+1
          do k=1,7
            frac(k,i)=frac(k,ifr)
           enddo
          frac(3,ifr)=frac(3,ifr)*c
          inew(ifr)=i
         else
          c=0.
         endif
       enddo
      nfrac=i
      write(6,6102)nfrac
 6102 format(' You have selected ',i6,' fractures ')
      do ifr=nfrac,1,-1
        i=2*ifr-1
        do k=1,7
          frac(k,i)=frac(k,ifr)
         enddo
        f1=frac(1,i)*pi180
        f2=frac(2,i)*pi180
        trig(1,i)=cos(f1)
        trig(2,i)=sin(f1)
        trig(3,i)=cos(f2)
```

```
        trig(4,i)=sin(f2)
       enddo
      nfrac=2*nfrac
c
      i=0
      do iout=1,nout
        if(ifint(7,iout).ne.0)then
          i=i+1
          do k=1,6
            end(k,i)=end(k,iout)
           enddo
          do k=7,8
            ifi=ifint(k,iout)
            ifint(k,i)=inew(ifi)
           enddo
         endif
       enddo
      ndot=i
c
      do iint=nout+1,nout+nint
        if(ifint(7,iint).ne.0)then
          i=i+1
          do k=1,6
            end(k,i)=end(k,iint)
           enddo
          do k=7,8
            ifi=ifint(k,iint)
            if(ifi.le.0)then
              ifint(k,i)=ifi
             else
              ifint(k,i)=inew(ifi)
             endif
           enddo
         endif
       enddo
      nseg=i
      close(unit=1)
      return
   60 call donepl
      stop ' end of last data file '
c
 1001 format(20a4)
 1002 format(7f10.4)
 1003 format(3i10)
 1004 format(10i6)
 1005 format(a4)
 1006 format(2f10.4,i10)
 1007 format(6i10)
 1008 format(i6,7e10.3)
 1009 format(i6,6f10.4,2i10)
      end
c
c
c
      subroutine rotate(xyz,matrix)
c
c     compute [matrix]*(xyz)
c
      real matrix(3,3),xyz(3),a(3)
      do i=1,3
```

```
        a(i)=0.
        do j=1,3
          a(i)=a(i)+matrix(i,j)*xyz(j)
         enddo
       enddo
      do i=1,3
        xyz(i)=a(i)
       enddo
      return
      end
c
c
c
      subroutine rotfrac(ifrac,matrix)
c
c     compute the sines and cosines of the normal to fracture
c     # ifrac after rotation by [matrix].
c
      include 'common.dim'
      real matrix(3,3),xyz(3)
c
c     vector normal to fracture
c
      xyz(1)=trig(2,ifrac)*trig(3,ifrac)
      xyz(2)=-trig(2,ifrac)*trig(4,ifrac)
      xyz(3)=trig(1,ifrac)
c
c     rotate vector
c
      call rotate(xyz,matrix)
c
c     compute sine and cosine of new phi and theta
c
      if(xyz(3).lt.0.)then
        do i=1,3
          xyz(i)=-xyz(i)
         enddo
       endif
      trig(1,ifrac)=xyz(3)
      cmax=0.999999
      if(trig(1,ifrac).le.cmax)then
        trig(2,ifrac)=sqrt(1.-trig(1,ifrac)*trig(1,ifrac))
        trig(3,ifrac)=xyz(1)/trig(2,ifrac)
        trig(4,ifrac)=-xyz(2)/trig(2,ifrac)
       else
        trig(2,ifrac)=0.
        trig(3,ifrac)=1.
        trig(4,ifrac)=0.
       endif
      return
      end
c
c
c
      subroutine rotmat(phi,theta,matrix)
c
c     compute rotation matrix for angles phi and theta
c
      real matrix(3,3)
      parameter (pi180=0.0174532925)
```

```
cphi=cos(phi*pi180)
sphi=sin(phi*pi180)
ctheta=cos(theta*pi180)
stheta=sin(theta*pi180)
matrix(1,1)=cphi*ctheta
matrix(1,2)=cphi*stheta
matrix(1,3)=-sphi
matrix(2,1)=-stheta
matrix(2,2)=ctheta
matrix(2,3)=0.
matrix(3,1)=sphi*ctheta
matrix(3,2)=sphi*stheta
matrix(3,3)=cphi
return
end
```

This report was done with support from the Department of Energy. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of The Regents of the University of California, the Lawrence Berkeley Laboratory or the Department of Energy.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.

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