1/69

SOFTWARE RELEASE NOTICE

01. SRN Number: PA-SRN-165

02. Project Title: Part of Performance Assessment work. | Project No.: 20-1402-762

03. SRN Title: FAULTING Version 1.0

04. Originator/Requestor: Bruce Mabrito | Date: 01/21/98

05. Summary of Actions

- ☐ Release of new software
- ■ Release of modified software:
  - ■ Enhancements made
  - ☐ Corrections made
- ☐ Change of access software
- ☑ Software Retirement GW 11/30/2009

06. Persons Authorized Access

Name	RO/RW	A/C/D
Amitava Ghosh	RW	
John Stamatakos	RO	A
Sitakanta Mohanty	RW	A
Ron Janetzke	RW	A

07. Element Manager Approval: RG Date: 1/21/98

08. Remarks: FAULTING Version 1.0 has been incorporated into the TPA Code Version 3.0 and later versions as "FAULTO".

CNWRA Form TOP-6 (06/95)

2/65

SOFTWARE SUMMARY FORM

01. Summary Date: 01/16/98	02. Summary prepared by (Name and phone) B. Mabrito (210)522-5149	03. Summary Action: Recap
04. Software Date: 01/16/98	05. Short Title: FAULTING Version 1.0	
06. Software Title: FAULTING - Code for Simulation of Direct Fault Disruption, Version 1.0		07. Internal Software ID: None

08. Software Type:	09. Processing Mode:	10. Application Area
☐ Automated Data System	■ Interactive	A. General:
■ Computer Program	☐ Batch	☐ Scientific/Engineering ☐ Auxiliary Analyses
☐ Subroutine/Module	☐ Combination	☐ Total System PA
		■ Subsystem PA ☐ Other
		B. Specific:

11. Submitting Organization and Address: CNWRA/SwRI 6220 Culebra Road San Antonio, TX 78238	12. Technical Contact(s) and Phone: Amitava Ghosh (210)522-3314

13. Narrative: The FAULTING Code simulates the effects of fault slip on waste package disruption in the proposed repository at Yucca Mountain.

14. Computer Platform: Sun Workstation	15. Computer Operating System: UNIX	16. Programming Language(s): FORTRAN 77	17. Number of Source Program Statements: approx. 5,000 lines
18. Computer Memory Requirements: Unknown	19. Tape Drives: None	20. Disk/Drum Units: N/A	21. Graphics: None; although creates the data file for Plot MTV program.

22. Other Operational Requirements: None

23. Software Availability:	24. Documentation Availability:
■ Available ☐ Limited ☐ In-House ONLY	■ Available ☐ Inadequate ☐ In-House ONLY

This documentation is being completed after the fact. Work on FAULTING has been finished for some time.

Software Custodian: Date: 1/21/98

CNWRA Form TOP-4-1

CENTER FOR NUCLEAR WASTE REGULATORY ANALYSES
QUALITY ASSURANCE
SURVEILLANCE REPORT

PROJECT NO.: 20.01402.159	REPORT NO.: 2000-13	PAGE 1 OF 2

SURVEILLANCE SCOPE: Review of CNWRA Developed Scientific and Engineering Software to determine whether the documentation present in the CNWRA Software Working Records Folders is adequate.

REFERENCE DOCUMENTS: Technical Operating Procedure-018, Development and Control of Scientific and Engineering (S&E) Software; QAP-004, Surveillance Control; Nonconformance Report 2000-03.

STARTING DATE: 3/7/2000 ENDING DATE: 6/9/2000

QA REPRESENTATIVE: B. Mabrito

PERSONS CONDUCTING TEST/EXAM/ACTIVITY: Various CNWRA staff working on Developed S&E software.

SATISFACTORY FINDINGS: During the course of this surveillance, CNWRA Developed S&E software and documentation was checked and contact made with CNWRA staff who worked with the software. In each case, the particular S&E software folder was reviewed for completeness and where no Design Verification Report (DVR) was located, the objective evidence in the folder was compared to the DVR form questions and discussions were held with cognizant CNWRA staff. The list of Developed S&E software reviewed is included in Attachment A.

In each case, key elements of the DVR were compared against that which was included in each software folder in the QA working records. Also, the previous version of the software code documentation was checked to ensure that the earlier DVR had been properly completed. The later version of the software documentation showed the specific changes made through the Software Change Reports. Based on this review, it is clear that although in a few cases no DVR was accomplished, product quality did not suffer. The minor enhancements and "bug" fixes made to TPA Version 3.2.3 and 3DStress Version 1.3.1 and 1.3.2 software were clearly identified and controlled so that the CNWRA product being delivered met the client's requirements.

UNSATISFACTORY FINDINGS: None.

NONCONFORMANCE REPORT NO.: None.

ATTACHMENTS: Attachment A.

RECOMMENDATIONS/ACTIONS: N/A.

APPROVED: 
CENTER DIRECTOR OF QUALITY ASSURANCE

DATE: 6/12/2000

DISTRIBUTION:
ORIGINAL - CENTER QA DIRECTOR QA Records
ORIGINATOR
PRINCIPAL INVESTIGATORS OF EACH CODE
ELEMENT MANAGERS
B. Sagar, H. Garcia

ATTACHMENT A

NAME OF S&E SOFTWARE	DESIGN VERIFICATION REPORT		NOTES
3DStress Version 1.2	Present	Dated 5/8/97	
3DStress Version 1.3	Present	Dated 8/7/98	
3DStress Version 1.3.1	Not Present		Software Release Notice Dated 7/15/99
3DStress Version 1.3.2	Not Present		Software Release Notice Dated 9/16/99
ASHPLUME Version 1.0	Present	Dated 6/23/97	
BREATH Version 1.1	Not Present		Software Release Notice Dated 9/21/95
BREATH Version 1.2	Present	Dated 9/17/97	
EBSPAC Version 1.0	Present	Dated 5/15/97	
EBSPAC Version 1.1	Present	Dated 6/17/97	
FAULTING Version 1.0	Not Present		Software Release Notice Dated 1/21/98 Module put under TPA Code and controlled in that manner.
GEOINVRT Version 1.0	Software Code Not Finished		Software Requirements Description only.
HAZINFO Version 1.0	Software Code Not Finished		Software Requirements Description only.
MULTIFLO Version 1.2	Present	Dated 3/2/2000	
MULTIFLO Version 2.0	Software Code Not Finished		Software Requirements Description only.
PVHA Version 1.0	Present	Dated 2/15/2000	
SUFLAT Version 1.0	Not Present		Element Manager (EM) determined that this software has not been used for regulatory reviews and will not be used for such work. EM requested the folder be archived in QA Records to reflect previous efforts on code.
TECTRAN Version 1.0	Software Code Not Finished		Software Requirements Description only.
TPA Version 3.2	Present	Dated 7/17/98	
TPA Version 3.2 (PP) Beta	Present	Dated 11/25/98	
TPA Version 3.2.3	Not Present		Software Release Notice Dated 7/14/99
TPA Version 3.3	Present	Dated 11/24/99	
TPA Version 4.0	Present	Dated 3/31/2000	

4/65

MEMO:

TO: B. Mabrito, QA Director

FROM: A. Ghosh and R. G. Baca

DATE: April 30, 1997

RE: FAULTING code (Version 1.0)

CC: J. Stamatakos, L. M. McKague

The FAULTING code (version 1.0) was developed to study the effects of fault displacement on waste package performance in the proposed repository at Yucca Mountain. Initially, the code was developed as a stand-alone version and a user's manual (FAULTING VERSION 1.0 – A CODE FOR SIMULATION OF DIRECT FAULT DISRUPTION TECHNICAL DESCRIPTION AND USER'S GUIDE, A. Ghosh, R. D. Manteufel, and G. L. Stirewalt, CNWRA 97-002) was issued in January 1997. Later on a simplified version of the FAULTING code was implemented in the TPA Version 3.0 as the FAULTO module. Consequently, development of the FAULTING code as an independent code is not pursued at this point. Moreover, we plan to study the effects of faulting on performance through applications of the TPA code (i.e., not through applications of the standalone FAULTING module).

5/65

Faulting
Software Folder

To: Bruce Mabrito at CNWRA-OS2
To: Linda Hearon at CNWRA-OS2
CC: Amitava Ghosh
CC: Asadul Chowdhury at CNWRA-OS2
CC: Henry Garcia at CNWRA-OS2
From: Pat Starkweather
Subject: Version Control of Faulting
03-10-97 10:29 AM

Mr. Mabrito; 10 Mar 97

Herewith notification that an S&E Software code has been placed under Version Control in accordance with TOP-018.

Details

Program Name: Faulting
Date Entered: 7 Mar 97
Control Method: SCCS
Version Nr: 1.0
Location: mammoth:/lan/rcs/faulting

This code consists of one f77 file, fault.f; and one include file, fault.inc

In addition to these two files, the WordPerfect file associated with faulting is also in this subdirectory under '/lan/rcs/faulting/doc/fault.wp5' and an associated tex (laTex) file is also available.

A single 3 1/2" floppy diskette was sufficient to hold all of the above, and was created, labeled, and given to L.Hearon for inclusion in the faulting package.

I did ***NOT*** run any software consistency checking tools or do any installation testing of faulting.

Pat Starkweather
x-5238

UNITED STATES
NUCLEAR REGULATORY COMMISSION
WASHINGTON, D.C. 20555-0001

6/64

February 18, 1997

SUBJECT CODE 102.2
PROJECT NO. 20-5708-762

Dr. Robert G. Baca
Performance Assessment Element Manager
Center for Nuclear Waste Regulatory Analyses
6220 Culebra Road
San Antonio, Texas 78228-0510

Dear Dr. Baca:

SUBJECT: ACCEPTANCE OF INTERMEDIATE MILESTONE: 5708-762-700, FAULTING MODULE LETTER REPORT

We have reviewed and accept the CNWRA Letter Report documenting the completion of the faulting module developed for inclusion into the Total System Performance Assessment (TPA) 3.0 code. We acknowledge that this code currently is in a stand-alone form, but will be incorporated into the TPA 3.0 code by the March 17, 1997, deliverable date for the system code. As part of that incorporation, sampling of parameters, currently performed in the Faulting Module, will be removed and all sampling as part of TPA 3.0 will be done at the executive level.

If you have any questions, please contact me at (301) 415-7289.

Sincerely,

Keith I. McConnell, Program Element
Manager for Geology/Geophysics
Geologic Setting Program Element
Division of High-Level Waste Management
Office of Nuclear Materials Safety
and Safeguards

cc: B. Meehan, CAB1/ADM
J. Linehan, PMDA

cc: W. Patrick
Directors
Element Managers
S. Mohanty

5/69

FAULTING CONSEQUENCE MODULE CODE SOFTWARE REQUIREMENTS DESCRIPTION

By
Amitava Ghosh

Center for Nuclear Waste Regulatory Analyses
San Antonio, Texas

Reviewed by:

Randall D. Manteufel 12/10/96
Randall D. Manteufel

Approved by:

RG Baca 12/11/96
Robert G. Baca, Manager, Performance Assessment
CNWRA

8/69

FAULTING CONSEQUENCE MODULE CODE SOFTWARE REQUIREMENTS DESCRIPTION

1 INTRODUCTION

This software requirements description document is the first step in construction of the FAULTING code which is intended to be used in the Nuclear Regulatory Commission/Center for Nuclear Waste Regulatory Analyses (NRC/CNWRA) Iterative Performance Assessment (IPA). The FAULTING code is a stamd-alone code that can also be used as a consequence module for the Total Performance Assessment (TPA) code. The TPA code simulates the performance of a geologic repository of nuclear high-level waste (HLW) at Yucca Mountain (YM), Nevada.

2 SOFTWARE FUNCTION

Using variables based on published field data, the module generates a faulting event in a simulation area measuring 50×50 km centered around the potential repository block at YM, with displacement assumed to occur along an unknown, randomly-located fault zone which may develop during the 10,000-yr regulatory time frame of interest. The module uses published field data for simulating timing and amount of both largest credible and cumulative types of displacement. The FAULTING module provides a framework for determining if primary fault displacement in the repository block could induce waste package (WP) disruption. The code determines the number of WPs disrupted and the timing of that disruption if it occurs. The FAULTING consequence module will permit an assessment of fault displacement hazards in the repository block.

3 TECHNICAL AND COMPUTATIONAL APPROACH

Fault displacement is generated in the FAULTING module along an assumed, unknown, randomly located fault zone inside the simulation area. These unknown fault zones include those not distinguished or adequately characterized, as well as new faults which may develop during the regulatory time frame of interest. Although the time frame considered is 10,000 yr, the approach is amenable to analysis over longer periods should the need arise. Strike direction is determined as either northwest or northeast parallel to the fault trace orientations observed in the field at and near YM. Whether the fault intersects the potential repository depends on location and orientation of the fault in the simulation area and total fault trace length. A WP disruption threshold value is governed by repository and waste package design and waste package emplacement geometry. If the threshold displacement is exceeded by either largest credible displacement in a single event or by smaller cumulative displacements through multiple events over time, the number and locations of WPs intersected and disrupted are calculated.

It is assumed that the unknown fault zones generally possess attributes similar to those of the Ghost Dance and Sundance faults, which have been mapped in the repository block. Using published field data the following variables for defining the fault zone are chosen randomly from ranges of values which are represented as probability distribution functions (PDFs): location; trace orientation; geometry (fault length, dip, and width); fault activity; and time and magnitude of largest credible displacement faulting events and cumulative displacement rate. The fault zone is assigned a randomly selected width, and displacement in the zone is considered along single or multiple slip surfaces, if desired. Because faults

are observed to dip steeply (i.e., between 60 and 90°) at the surface and similar dips are thought to occur at repository level, it is assumed that variation in dip of the fault has little influence on number of WPs disrupted, considering horizontal WP emplacement. If a largest credible displacement event or cumulative displacement exceeds the WP disruption threshold, a WP disruption occurs. The timing and the number of WPs disrupted are then calculated based on the repository WP loading density.

A more complete discussion of the technical basis and computational approach for this module is described in Stirewalt et. al., (1995).

4 USER INTERFACE AND DATA FLOW

The code will be designed to accept input data for parameter values and distributions from a file and will write the results of the calculations to an output file.

4.1 INPUT TO FAULTING CODE

The code receives input data for parameter values and distributions from fault.inc file. Information on the size and location of the repository regions are contained in the source code. Initially the module will take the required input data on which repository region is of interest and on the desired number of realizations through the standard input (i.e., keyboard) during runtime. No other files/input will have an effect on the results of the calculation.

4.2 OUTPUT FROM FAULTING CODE

The FAULTING simulations produce output files which list those faults which intersect the repository. Output data listed includes fault size and orientation information, largest credible event displacement, threshold displacement, and lists area, number, and percentage of WP disrupted. FAULTING also generates a file for plotting the fault traces, if desired.

5 PROGRAMMING LANGUAGES

The FAULTING code is written in standard FORTRAN 77 as implemented in with SUN SPARCompiler, Version 2.0.

6 HARDWARE PLATFORMS

The FAULTING code will be developed for execution on SUN machines using the UNIX operating systems.

7 GRAPHICS OUTPUT

No graphics output is required or supported by the FAULTING code.

10/69

8 PRE AND POST-PROCESSOR

No pre or post-processor is required or supported by the FAULTING code. The output files generated by the code will be designed so that they can be read easily by spreadsheet programs, analysis software, and plotting packages.

9 MATHEMATICAL MODEL, CONTROL FLOW, DATA FLOW, CONTROL LOGIC, AND DATA STRUCTURE

Please refer to the CNWRA report by Stirewalt et al., (1995).

10 REFERENCES

Stirewalt, G.L., S.M. McDuffie, R.D. Manteufel, and R.W. Janetzke. 1995. Technical Specification for a Fault Displacement Module. Report to the Nuclear Regulatory Commission, San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.

DRAFT

FAULTING CONSEQUENCE MODULE CODE SOFTWARE REQUIREMENTS DESCRIPTION

11/69

By
Amitava Ghosh

Center for Nuclear Waste Regulatory Analyses
San Antonio, Texas

Reviewed by:

_____

Randall D. Manteufel

Approved by:

_____

Robert G. Baca, Manager, Performance Assessment
CNWRA

DRAFT

FAULTING CONSEQUENCE MODULE CODE SOFTWARE REQUIREMENTS DESCRIPTION

12/69

1 INTRODUCTION

This software requirements description document is the first step in construction of the FAULTING code which is intended to be used in the Nuclear Regulatory Commission/Center for Nuclear Waste Regulatory Analyses (NRC/CNWRA) Iterative Performance Assessment (IPA). The FAULTING code is a stamd-alone code that can also be used as a consequence module for the Total Performance Assessment (TPA) code. The TPA code simulates the performance of a geologic repository of nuclear high-level waste (HLW) at Yucca Mountain (YM), Nevada.

2 SOFTWARE FUNCTION

Using variables based on published field data, the module generates a faulting event in a simulation area measuring 50×50 km centered around the potential repository block at YM, with displacement assumed to occur along an unknown, randomly-located fault zone which may develop during the 10,000-yr regulatory time frame of interest. The module uses published field data for simulating timing and amount of both largest credible and cumulative types of displacement. The FAULTING module provides a framework for determining if primary fault displacement in the repository block could induce waste package (WP) disruption. The code determines the number of WPs disrupted and the timing of that disruption if it occurs. The FAULTING consequence module will permit an assessment of fault displacement hazards in the repository block.

3 TECHNICAL AND COMPUTATIONAL APPROACH

Fault displacement is generated in the FAULTING module along an assumed, unknown, randomly located fault zone inside the simulation area. These unknown fault zones include those not distinguished or adequately characterized, as well as new faults which may develop during the regulatory time frame of interest. Although the time frame considered is 10,000 yr, the approach is amenable to analysis over longer periods should the need arise. Strike direction is determined as either northwest or northeast parallel to the fault trace orientations observed in the field at and near YM. Whether the fault intersects the potential repository depends on location and orientation of the fault in the simulation area and total fault trace length. A WP disruption threshold value is governed by repository and waste package design and waste package emplacement geometry. If the threshold displacement is exceeded by either largest credible displacement in a single event or by smaller cumulative displacements through multiple events over time, the number and locations of WPs intersected and disrupted are calculated.

It is assumed that the unknown fault zones generally possess attributes similar to those of the Ghost Dance and Sundance faults, which have been mapped in the repository block. Using published field data the following variables for defining the fault zone are chosen randomly from ranges of values which are represented as probability distribution functions (PDFs): location; trace orientation; geometry (fault length, dip, and width); fault activity; and time and magnitude of largest credible displacement faulting events and cumulative displacement rate. The fault zone is assigned a randomly selected width, and displacement in the zone is considered along single or multiple slip surfaces, if desired. Because faults

are observed to dip steeply (i.e., between 60 and 90°) at the surface and similar dips are thought to occur at repository level, it is assumed that variation in dip of the fault has little influence on number of WPs disrupted, considering horizontal WP emplacement. If a largest credible displacement event or cumulative displacement exceeds the WP disruption threshold, a WP disruption occurs. The timing and the number of WPs disrupted are then calculated based on the repository WP loading density.

A more complete discussion of the technical basis and computational approach for this module is described in Stirewalt et. al., (1995).

4 USER INTERFACE AND DATA FLOW

The code will be designed to accept input data for parameter values and distributions from a file and will write the results of the calculations to an output file.

4.1 INPUT TO FAULTING CODE

The code receives input data for parameter values and distributions from fault.inc file. Information on the size and location of the repository regions are contained in the source code. Initially the module will take the required input data on which repository region is of interest and on the desired number of realizations through the standard input (i.e., keyboard) during runtime. No other files/input will have an effect on the results of the calculation.

4.2 OUTPUT FROM FAULTING CODE

The FAULTING simulations produce output files which list those faults which intersect the repository. Output data listed includes fault size and orientation information, largest credible event displacement, threshold displacement, and lists area, number, and percentage of WP disrupted. FAULTING also generates a file for plotting the fault traces, if desired.

5 PROGRAMMING LANGUAGES

The FAULTING code is written in standard FORTRAN 77 as implemented in with SUN SPARCompiler, Version 2.0.

6 HARDWARE PLATFORMS

The FAULTING code will be developed for execution on SUN machines using the UNIX operating systems.

7 GRAPHICS OUTPUT

No graphics output is required or supported by the FAULTING code.

DRAFT 14/69

8 PRE AND POST-PROCESSOR

No pre or post-processor is required or supported by the FAULTING code. The output files generated by the code will be designed so that they can be read easily by spreadsheet programs, analysis software, and plotting packages.

9 MATHEMATICAL MODEL, CONTROL FLOW, DATA FLOW, CONTROL LOGIC, AND DATA STRUCTURE

Please refer to the CNWRA report by Stirewalt et al., (1995).

10 REFERENCES

Stirewalt, G.L., S.M. McDuffie, R.D. Manteufel, and R.W. Janetzke. 1995. Technical Specification for a Fault Displacement Module. Report to the Nuclear Regulatory Commission, San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.

Faulting Folder

15/69

[63] From: Randall Manteufel at CNWRA 12/6/96 4:16PM (1096 bytes: 1 ln)
To: Bruce Mabrito at CNWRA-OS2
cc: Mark Jarzemba, Amitava Ghosh at CNWRA-SUN, Robert Baca at CNWRA-OS2
Subject: SRD's
---------- Message Contents ----------
Bruce,

We have decided to develop stand-alone SRDs for the ASHPLUME & FAULTING computer programs given their 1) extensive use at the Center, 2) recognition by the NRC staff, and 3) anticipated future use in Center deliverables.

Earlier, we had planned to include descriptions of these 2 codes in the TPA SRD. But now we will issue individual SRDs.

You will receive 3 SRDs for the following codes:

1) FAULTING (lead: Amit Ghosh) by 12/15/96
2) ASHPLUME (lead: Mark Jarzemba) by 12/15/96
3) TPA 3.0 (lead: Randy Manteufel) by 1/15/97

Individual User Guides will also be completed for each of the 3 codes. Anticipated dates are:

1) FAULTING (lead: Amit Ghosh) by 1/20/97
2) ASHPLUME (lead: Mark Jarzemba) by 2/97 (?)
3) TPA 3.0 (lead: Randy Manteufel) by 3/17/97

Thanks.

-Randy

SCIENTIFIC NOTEBOOK
Development of the FAULTING Module

by

Amitava Ghosh

Southwest Research Institute
Center for Nuclear Waste Regulatory Analyses
San Antonio, Texas

January 20, 1997

Table of Contents

List of Figures

Printed: February 7, 1997

Amitava Ghosh SCIENTIFIC NOTEBOOK INITIALS: AG

1. INITIAL ENTRIES

Scientific NoteBook: # 167

Issued to: Amitava Ghosh

Issue Date: February 28, 1996

Printing Period: October 1 through January 31, 1996

Project Title: Development of the FAULTING Module of Phase 3 IPA

Project Staff: Amitava Ghosh

By agreement with the CNWRA QA this NoteBook is to be printed at approximate quarterly intervals. This computerized Scientific NoteBook is intended to address the criteria of CNWRA QAP-001.

[Amitava Ghosh, February 28, 1996]

1.1. Objectives

The objective of this activity is to develop the FORTRAN coding of the faulting consequence (FAULTING) module for the Phase 3 of the Nuclear Regulatory Commission (NRC) Iterative Performance Assessment (IPA) activity. The basic approach to the FAULTING module has been described in Stirewalt et al. (1995). Taking into account the published data for faults at Yucca Mountain area, this module provides a framework for determining if the primary fault displacement (that is, displacement along the main fault trace rather than along associated secondary fractures) in the repository block could induce waste package disruption, the timing of that disruption, and the number of waste packages disrupted. [Amitava Ghosh, February 28, 1996]

1.2. Technical Approaches

The codes for the FAULTING module will be written in Fortran 77. Fault displacement is generated in the FAULTING module along an assumed, unknown, and randomly located fault zone inside the simulation area. These unknown fault zones include those not distinguished or adequately characterized, as well as new faults which may develop during the 10,000 year regulatory time frame of interest. Strike direction is determined as either northeast or northwest observed in the field at and near Yucca Mountain. Whether the fault intersects the potential repository depends on location and orientation of the fault in the simulation area and total fault trace length. Whether waste packages are disrupted is dependent upon amount of displacement exceeding a threshold value which is governed by repository and waste package design and waste package emplacement geometry. If the threshold displacement is exceeded by either

largest credible displacement in a single event or by smaller cumulative displacements with time through multiple events, then number and locations of waste packages intersected and disrupted can be calculated based on length of intersection of the fault zone with the repository, repository design, and waste package emplacement geometry. Location, trace orientation, geometry, activity, number and time and amount of largest credible displacement faulting events, and amount and time of cumulative displacements are chosen randomly from range of published data represented as probability distribution functions (PDFs). It is assumed that the unknown fault zones can possess attributes similar to those of the Ghost Dance and Sundance faults. The fault zone is assigned a randomly selected width, and displacement in the zone is considered along both single and multiple slip surfaces. It is assumed that the variation in dip of the fault has little influence on number of waste packages disrupted, considering horizontal emplacement. If waste package disruption occurs as a result of either a largest credible displacement event or cumulative displacement, the timing of that disruption is relayed to the SOurce TErm Code (SOTEC) for calculation of radionuclide release.

The random numbers following a particular PDF will be generated primarily using the program LHS94 (version 1.1) (Iman and Shortencarier, 1984) which uses the Latin Hypercube method. However, using the LHS94 program is a two step process. All the random numbers needed to run the FAULTING modules should be known beforehand and need to generated in advance of running the module. The final version of the FAULTING module will have an user interface to do this. Additionally, each subroutine of the FAULTING module may use own random number generation routines during development. Currently, the LHS94 code is not under the code configuration management of CNWRA.

[Amitava Ghosh, February 28, 1996]

The code in the present form has own subroutines to generate necessary random numbers from a given distribution. Therefore, at this point, there is no need to generate the random numbers using the LHS routines before the run and use them while running this module. At present, this is quite a cumbersome process as we have to estimate beforehand how many random numbers from each specific distribution is needed. It may also be extremely difficult to guess beforehand as the number of random numbers needed for any simulation drastically goes down if the generated fault does not intersect the given repository region.

[Amitava Ghosh, June 27, 1996]

It was also decided to carry out the complete calculation in double precision. The reason is as follows. In the calculation done by this module, we are dealing with very large numbers (coordinates of the faults) and very small numbers (slipping rate on a fault surface). In both cases, it becomes necessary to keep enough precision in the calculations (at least more than 6 significant digits) so that rounding off error does not mislead the calculation sequence.

[Amitava Ghosh, June 27, 1996]

1.3. Computers, Computer Codes, and Data Files

Table 1-1. Computer, operating system, and compiler to be used in developing FAULTING module

Machine Name	Machine Type	Operating System	Compiler	Location
snowwhite	SUN SPARCstation 20	SUNOS 4.1	f77 (SUN)	Building 189

[Amitava Ghosh, February 28, 1996]

1.4. Mathematical Theory

Not Applicable

[Amitava Ghosh, February 28, 1996]

Two modifications of the algorithm given in Stirewalt et al. (1995) were carried out in this period:

1. Flow chart for calculation of largest credible displacement along single and multiple slip surfaces given in Stirewalt et al. (1995) has been modified to (i) reduce unnecessary calculations, and (ii) carry out the sequence of calculations in a more logical way.

Previously, after randomly generating the largest credible displacement, this amount of displacement was partitioned among 2 to 4 randomly generated slip surfaces of the same fault plane. Thereafter, the slip along each of the multiple surfaces was compared against the threshold value. If failure occurs on any of the multiple slip surfaces of the fault plane, the check for slippage on the fault plane itself. This procedure has been modified to allow the flow of calculation in a more logical way and to eliminate unnecessary calculations.

In the present procedure, once the largest credible displacement is randomly generated for a given fault plane, the displacement is compared with the threshold displacement TD for waste packages, based on a uniform pdf varying between 0.1 and 0.5 m. If only failure is indicated, the total displacement on this fault plane is partitioned among 2 to 4 slip surfaces generated randomly. Displacement on each of these slip surfaces is compared with the threshold displacement TD.

New flow chart is given in Figure 1-1. The code necessary to determine the number of waste packages being disrupted due to the faulting event is under development.

[Amitava Ghosh, August 18, 1996]

2. Previously, comparison of displacement along any of the slip surfaces was carried out against LDmax, which was set to 0.045 m. This is not correct as the correct comparison should be against the threshold displacement for waste package rupture, TD. The code has been modified to reflect this.

[Amitava Ghosh, August 18, 1996]

Algorithm to calculate fault displacement along multiple surfaces has been changed again after consultation with Drs. John Stamatokos and D. Ferrill. In the present algorithm, calculation for single and multiple surfaces are carried out independent of each other. Number of slip surfaces is simulated at the time of generating the fault geometrical parameters.

[Amitava Ghosh, November 15, 1996]

Previously, the threshold displacement for waste package disruption was a random variable varying uniformly between 0.1 m and 0.5 m. After discussion with Dr. B. Sagar, this has been changed to a user-specified variable. At the beginning of the simulation, user is asked interactively to input a specific value for this variable.

[Amitava Ghosh, November 20, 1996]

The new flow chart is given in Figure 1-2.

[Amitava Ghosh, January 28, 1997]

1.5. Assumptions

The assumptions for each of the parameters and the complete module are given in Stirewalt et al. (1995). [Amitava Ghosh, February 28, 1996]

1.6. Initial/Boundary Conditions

Not Applicable [Amitava Ghosh, February 28, 1996]

2. DEVELOPMENT OF THE FAULTING MODULE

2.1. Mathematical Theory

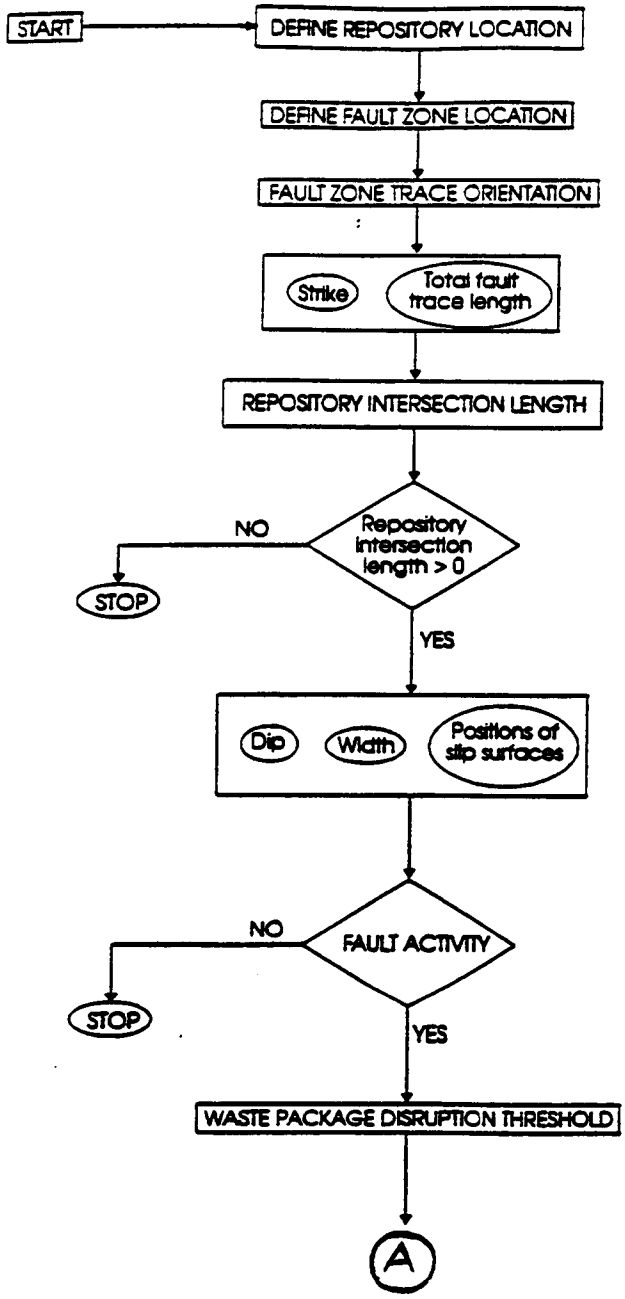

Figure 1-1. Flow diagram illustrating sequential steps for describing faulting events

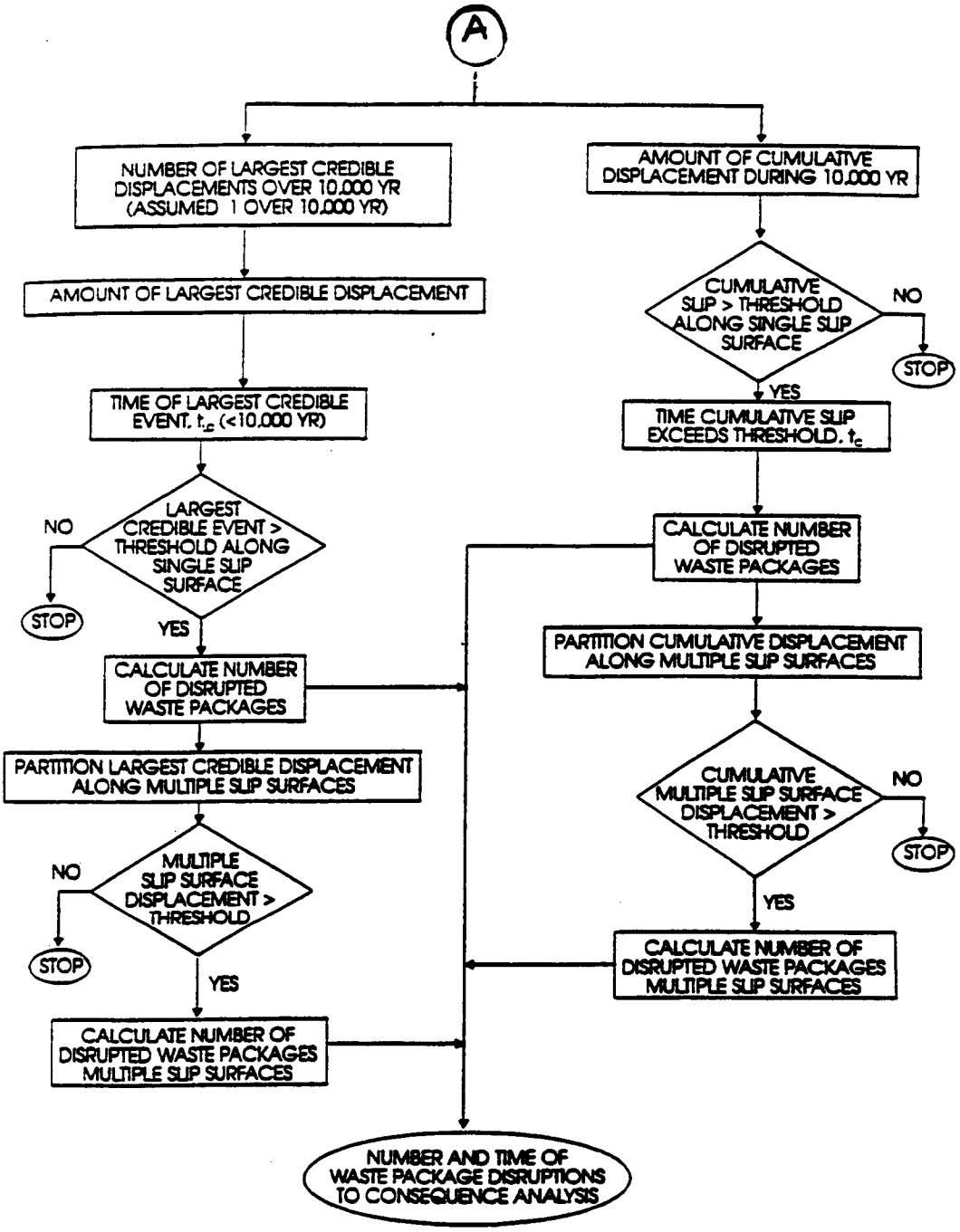

Figure 1-1. Flow diagram illustrating sequential steps for describing faulting events (cont.)

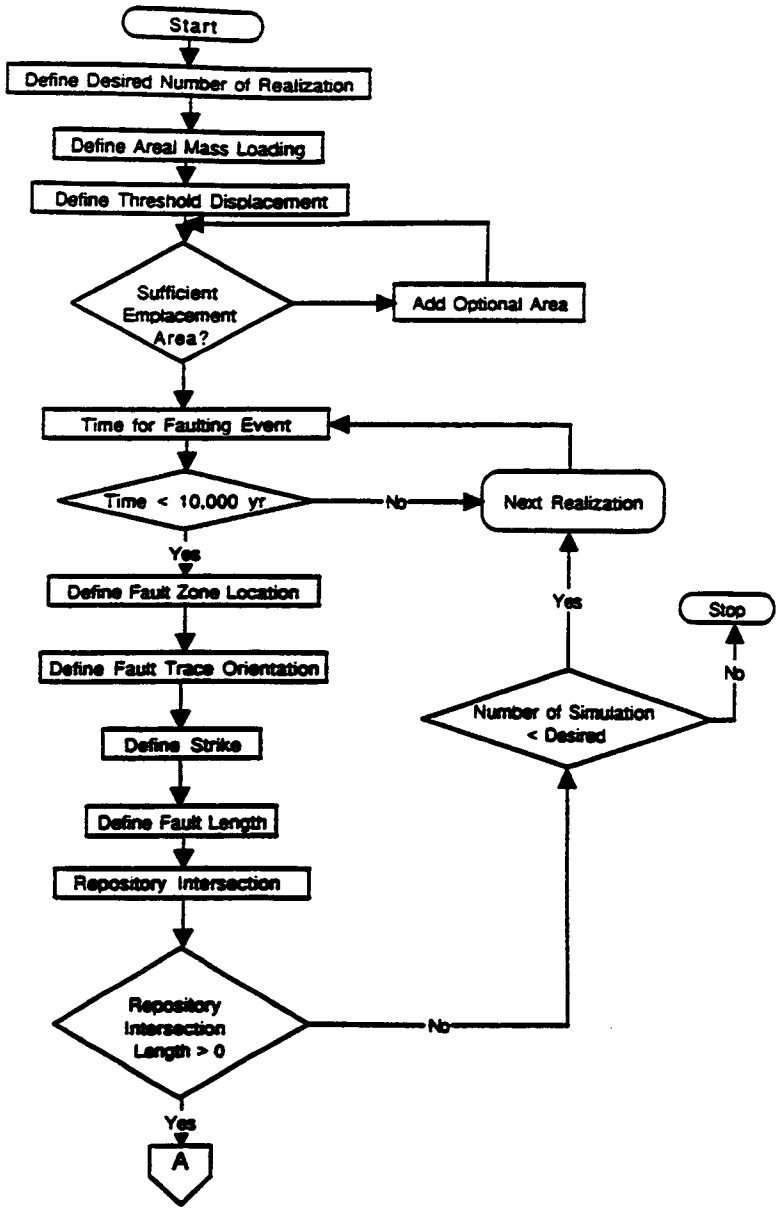

Figure 1-2. Flow diagram illustrating sequential steps for describing faulting events (revised)

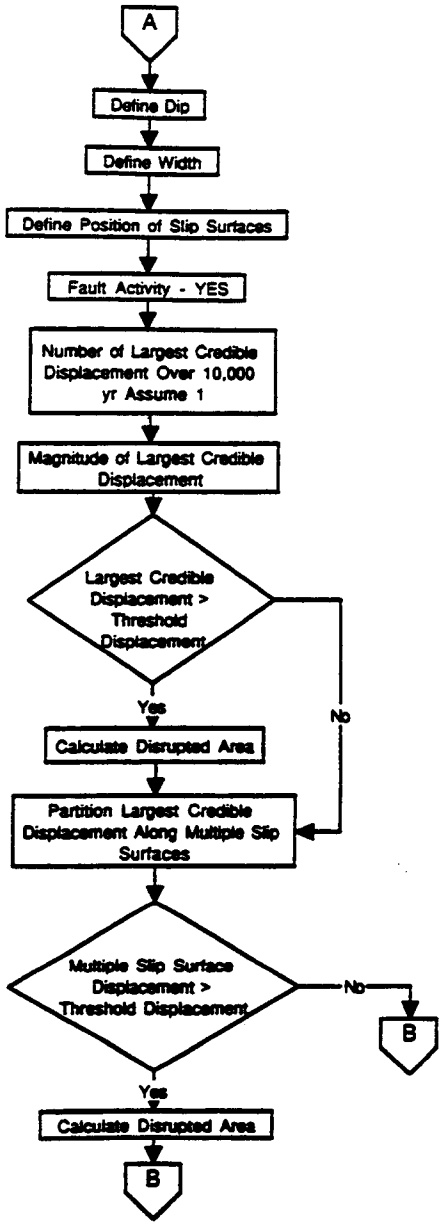

Figure 1-2. Flow diagram illustrating sequential steps for describing faulting events (revised) (cont.)

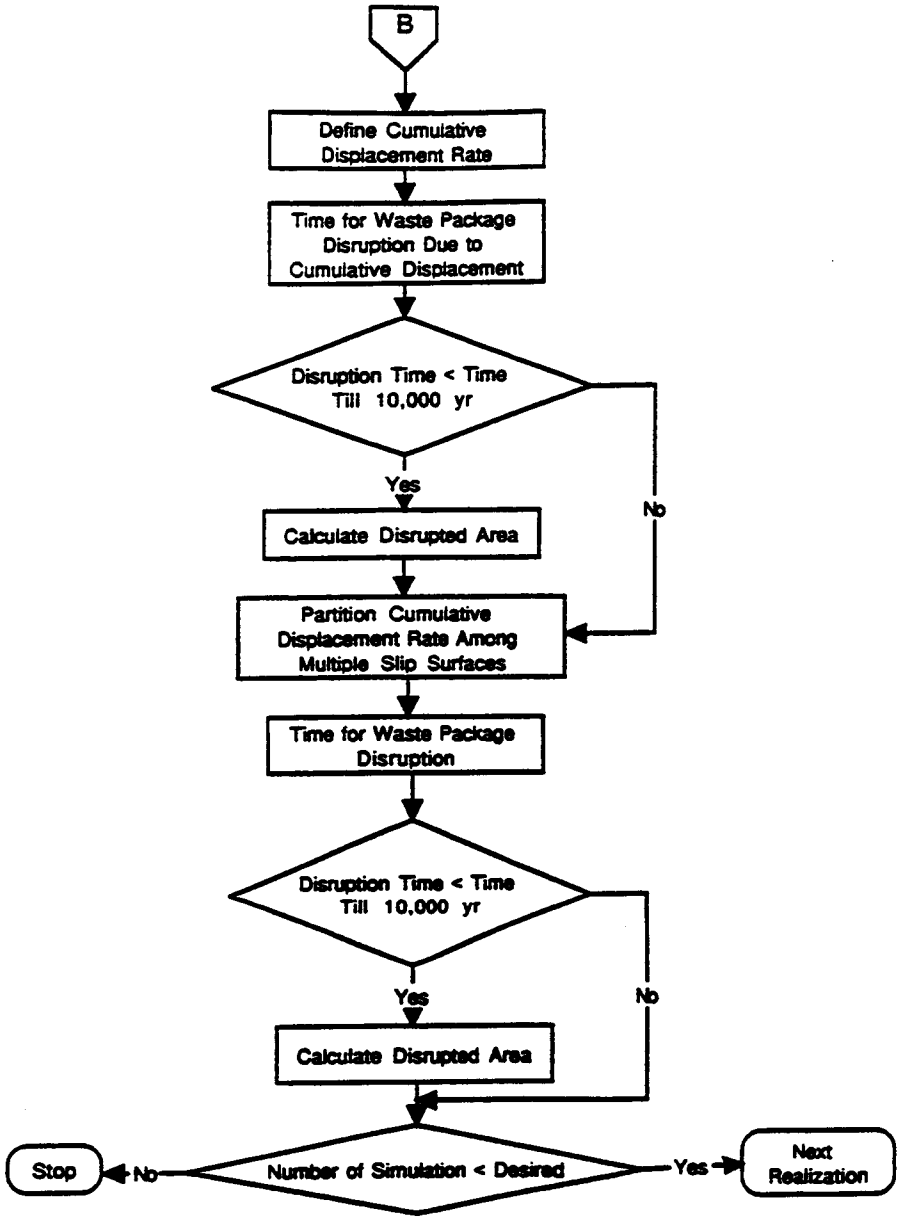

Figure 1-2. Flow diagram illustrating sequential steps for describing faulting events (revised) (cont.)

2.1.1. Coordinates of the Point of Intersection of Two Straight Lines Defined by the Coordinates of the End Points

Line 1: End points are (x_1,y_1) and (x_2,y_2)
Line 1: End points are (x_3,y_3) and (x_4,y_4)
Intersection point: (x_i,y_i)

Equation of line 1:

$$\frac{y-y_1}{x-x_1}=\frac{y_2-y_1}{x_2-x_1} \tag{1}$$

It goes through (x_i,y_i). Therefore,

$$\frac{y_i-y_1}{x_i-x_1}=\frac{y_2-y_1}{x_2-x_1} \tag{2}$$

Similarly,

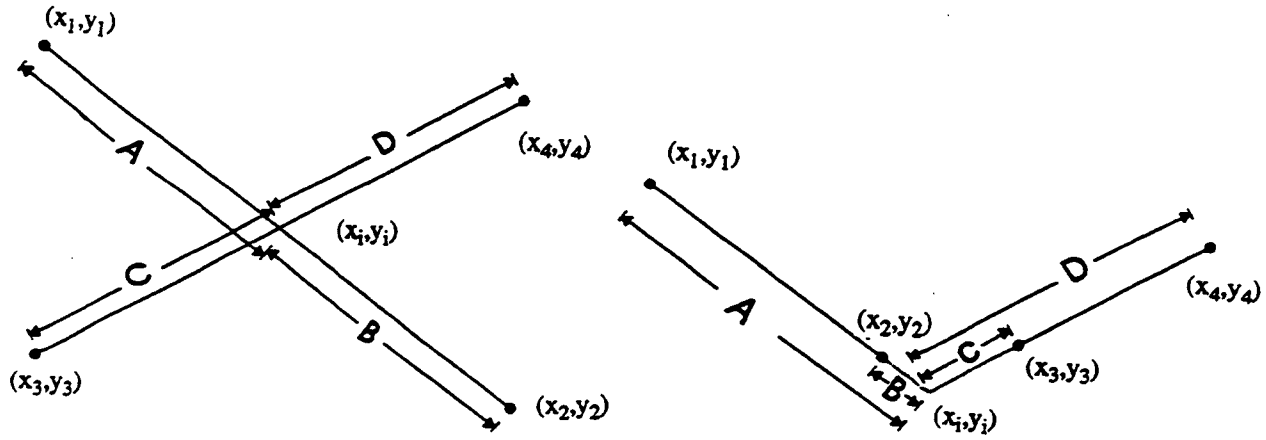

Figure 2-1. Determination of real intersection point of two finite straight lines

$$\frac{y_i - y_3}{x_i - x_3} = \frac{y_4 - y_3}{x_4 - x_3} \tag{3}$$

Solving (2) and (3) for x_i and y_i, we get

$$x_i = \frac{(x_1 - x_2)(x_3 y_4 - x_4 y_3) + (x_3 - x_4)(x_1 y_2 - x_2 y_1)}{-(x_3 y_1 - x_4 y_1 - x_3 y_2 + x_4 y_2 - x_1 y_3 + x_2 y_3 + x_1 y_4 - x_2 y_4)}$$

$$y_i = -\frac{x_2 y_1 y_3 + x_4 y_1 y_3 + x_1 y_2 y_3 - x_4 y_2 y_3 + x_2 y_1 y_4 - x_3 y_1 y_4 - x_1 y_2 y_4 + x_3 y_2 y_4}{-x_3 y_1 + x_4 y_1 + x_3 y_2 - x_4 y_2 + x_1 y_3 - x_2 y_3 - x_1 y_4 + x_2 y_4}$$

These solutions were also verified using the commercial program Mathematica. These equations were coded in the subroutine *intersection*.

These equations do not guarantee that the point of intersection of these lines are real, that is, the point lies on both the lines as shown:

A = length of the line joining (x_1,y_1) and (x_i,y_i)
B = length of the line joining (x_2,y_2) and (x_i,y_i)
C = length of the line joining (x_3,y_3) and (x_i,y_i)
D = length of the line joining (x_4,y_4) and (x_i,y_i)
E = length of the line joining (x_1,y_1) and (x_2,y_2)
F = length of the line joining (x_3,y_3) and (x_4,y_4)

For the intersection to be real, either of the following two conditions needs to be true.

$$A + B = E \rightarrow (A + B) - E = 0$$

or,

$$C + D = F \rightarrow (C + D) - F = 0$$

It should be recognized that comparison of real numbers having very similar magnitudes is problematic due to round off errors. Therefore, an acceptable tolerance should be used.

[Amitava Ghosh, March 31, 1996]

The formula for x_i has been changed to be parallel with that of y_i so that the coding of the subroutine becomes easier. The revised formula is as follows:

$$x_i = \frac{-x_2x_4y_3+x_1x_4y_3+x_2x_4y_1-x_2x_3y_1-x_1x_4y_2+x_1x_3y_2+x_2x_3y_4-x_1x_3y_4}{-x_3y_1+x_4y_1+x_3y_2-x_4y_2+x_1y_3-x_2y_3-x_1y_4+x_2y_4}$$

For the intersection to be real, both of the above two conditions needs to be true. Otherwise, in some cases wrong conclusions will be made. This has been corrected in the code.

[Amitava Ghosh, May 23, 1996]

2.1.2. Determination whether a given point falls within a quadrilateral defined by the four corners

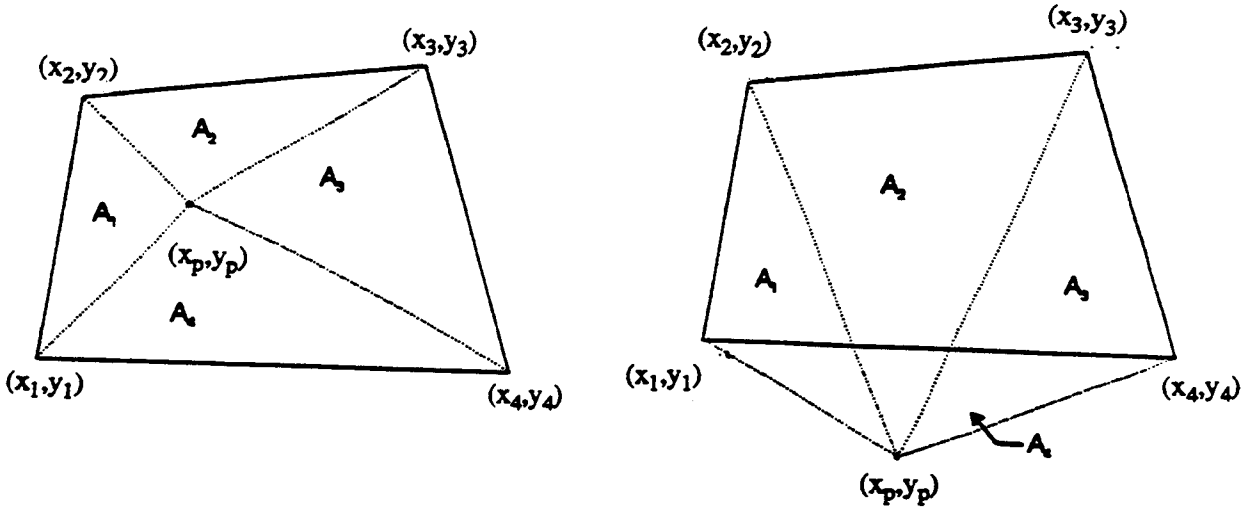

Figure 2-2. Determination whether a given point falls within a given quadrilateral

Given point is (x_p,y_p). The quadrilateral has corners defined by (x_1,y_1), (x_2,y_2), (x_3,y_3), and (x_4,y_4).

A1: area of triangle (x_1,y_1), (x_2,y_2), (x_p,y_p)
A2: area of triangle (x_2,y_2), (x_3,y_3), (x_p,y_p)
A3: area of triangle (x_3,y_3), (x_4,y_4), (x_p,y_p)

A4: area of triangle (x_4,y_4), (x_1,y_1), (x_p,y_p)

Area of quadrilateral (x_1,y_1), (x_2,y_2), (x_3,y_3), (x_4,y_4) = A
= area of [triangle (x_1,y_1), $x_2,y_2)$, (x_3,y_3) + triangle (x_3,y_3), (x_4,y_4) (x_1,y_1)]

If the point (x_p,y_p) is inside the quadrilateral, then

$$A = A_1 + A_2 + A_3 + A_4$$

If outside,

$$A < (A_1 + A_2 + A_3 + A_4)$$

[Amitava Ghosh, March 31, 1996]

2.1.3. Determination of the Fault Length Intersected by the Repository Block

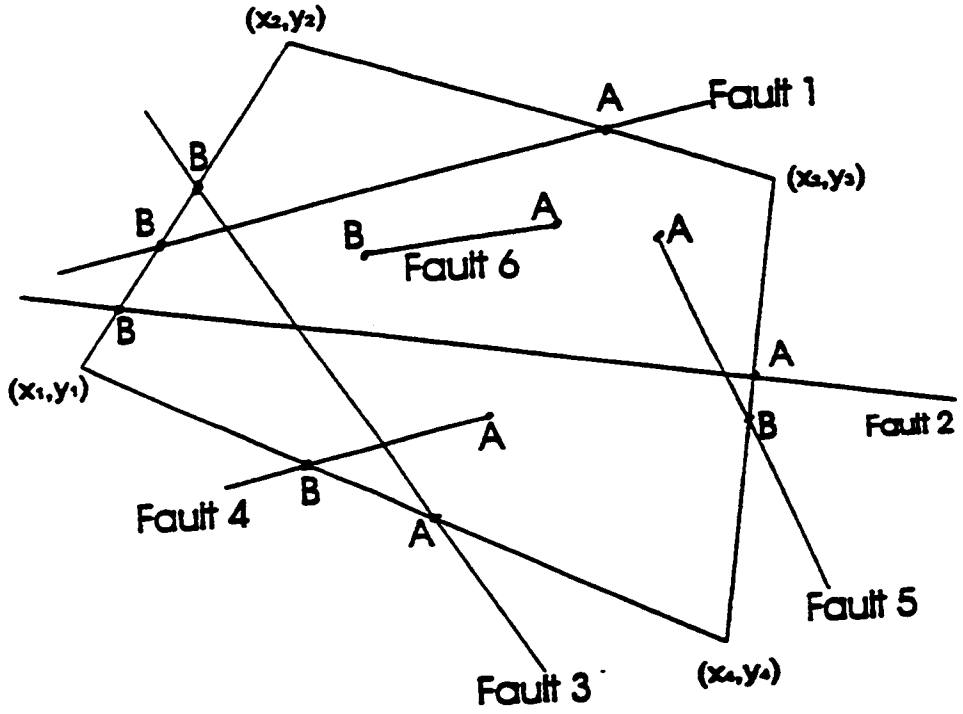

Figure 2-3. Determination of the length of a given fault intersected by a repository block

- Fault 7: Intersection length = Fault length
- Fault 5, Fault 6: Intersection length = AB: need to find only B
- Fault 1, Fault 2, Fault 3, Fault 4: Intersection Length = AB: need to know both A and B

Known: (i) Two ends of each fault (xf_1,yf_1) and (xf_2, yf_2)
(ii) Four corners (x_1,y_1), (x_2,y_2), (x_3,y_3), (x_4,y_4)

Step 1: Determine whether each end of the fault is within the boundary
if yes, intersection length = fault length

Step 2: If one end is within the boundary, determine
(i) which segment it is intersecting
(ii) calculate length AB

Step 3: If both ends are outside the boundary, determine
(i) which segments it is intersecting
(ii) calculate length AB

[Amitava Ghosh, March 31, 1996]

The length of the portion of a randomly generated fault intersected by a given repository region is calculated following the above-mentioned algorithm. Subroutine *checkin* is used to determine whether both ends of the fault fall within the given region. If both ends of the fault are within the given region then the intersected length is the length of the fault. If one end or both ends fall outside the given region, then subroutine *inlength* is called to determine first the intersection point(s) of the fault and the sides of the quadrilateral region. There will be one or two real intersection points depending on whether one or two end points fall outside the region, respectively. The intersected length for the first case is the distance between the real intersection point and the other end point of the fault which falls within the region. The intersected length for the second case is the distance between the two intersection points.

[Amitava Ghosh, June 7, 1996]

2.1.4. Random Numbers Having Normal Distribution with Mean μ and Standard Deviation σ, N (μ, σ_2)

Random numbers have to be generated with mean μ and standard deviation σ so that there is 90 percent probability that the number will fall between A and B.

$$\mu = \frac{A+B}{2}$$

$$P(x \le A) = \frac{100-90}{2}\% = 5\% = 0.05$$

$$P(X \ge B) = 0.05$$

$$P(Z \le 1.645) = 0.05$$

For a standard normal variate Z,

$$P(Z \le 1.645) = 0.05$$

Standard deviation of the desired truncated normal distribution is $\sigma = \frac{X-\mu}{1.645} = \frac{B-\mu}{1.645} = \frac{B-A}{2.29}$

Generation of normally distributed random numbers is carried out following Ang and Tang (1984):

If U_1 and U_2 are two uncorrelated uniformly distributed random numbers between 0 and 1, then two normally distributed random numbers with mean μ and standard deviation σ are:

$$x_1 = \mu + \sigma\sqrt{-2lnU_1}\,\mathrm{Cos}2\pi U_2$$

$$x_2 = \mu + \sigma\sqrt{-2lnU_1}\,\mathrm{Sin}2\pi U_2$$

[Amitava Ghosh, March 31, 1996]

Standard deviation of the desired truncated normal distribution is $\sigma = \frac{X-\mu}{1.645} = \frac{B-\mu}{1.645} = \frac{B-A}{3.29}$

This is the corrected form of the equation as 2x1.645 is equal to 3.29, not 2.29 as typed.

[Amitava Ghosh, May 21, 1996]

2.1.5. Beta Distribution in the Interval (A, B) with Parameters α and β

Random numbers Y from a general Beta distribution with lower and upper bounds A and B, respectively, may be obtained through

$$Y = \frac{X-A}{B-A}$$

where X is the corresponding random number for the standard Beta distribution

$$\mu_x = \frac{\alpha}{\alpha + \beta}$$

$$\sigma_x = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

Therefore, $\mu_y = (B-A)\mu_x + A = \frac{A\beta + B\alpha}{\alpha+\beta}$

$$\sigma_y = (B-A)^2\sigma_x = \frac{(B-A)^2\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

Generation of Beta distributed random numbers is carried out following Ang and Tang (1984):

A random number X with pdf of the standard Beta distribution with shape parameters σ and β can be generated from

$$X = \frac{U_1^{1/\alpha}}{U_1^{1/\alpha} + u_2^{1/\beta}}$$

where U_1 and U_2 are two random numbers having the standard uniform distribution provided $U_1^{1/\alpha} + U_2^{1/\beta} \le 1$.

[Amitava Ghosh, March 31, 1996]

2.1.6. LogBeta Distribution in the Interval (A, B) with Parameters α and β

X is standard beta distributed with mean equal to 0 and standard deviation equal to 1. Then Y has logbeta distribution if

$$Y = e^X$$

which means

$$\ln Y = X$$

has beta distribution. Consequently, the lower and upper limits will be:

$$a^* = \ln a$$

$$b^* = \ln b$$

Now, we have to generate a standard beta distribution with lower and upper limits are a^* and b^* respectively.

Mean of logbeta distribution $\mu = e^{\frac{\alpha}{\alpha+\beta}}$

Variance of logbeta distribution $\alpha = e^{\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}}$

Given $\alpha = 1.5, \quad \beta = 3.0$

$\mu = e^{\frac{1.5}{4.5}} = 1.3956$

$\sigma = e^{\frac{1.5x3}{4.5^2x(5.5)}} = 1.0412$

Minimum value $= e^0 = 1$

Maximum value $= e^1 = 2.71828$

Therefore, standard Beta x, i.e., within interval (0,1) distribution, when converted to logbeta distribution Y, will be within the interval (1, 2.71828)

$$Y = \frac{x-a}{b-a} = \frac{x-1}{1.71828}$$

Y will be logbeta distributed with interval between (0,1)

If, for example, we want random numbers which is logbeta within interval (A,B), then:

$$Z = Y(B-A)+A$$

$$Z = \frac{x-a}{b-a}\cdot(B-A)+A$$

$$= \frac{(x-a)(B-A)+A(b-A)}{b-a} = \frac{x(B-A)}{b-a} - \frac{a(B-A)}{b-a} + A$$

$$= x\frac{B-A}{b-a} + \frac{-aB+aA+bA-aA}{b-a}$$

$$= x\frac{B-A}{b-a} + \frac{bA-aB}{b-a}$$

[Amitava Ghosh, June 7, 1996]

2.1.7. Representation of Dip Angle

The given definition of dip given in Stirewalt et al. (1995) is quite inconvenient for computer processing. Dip direction and dip angle would have been a better alternative way of expressing the orientation of fault planes. Unfortunately, the data given here do not have correspondence between the strike and dip of a given fault plane. The field observation is restricted to the distribution of fault dip in two dominant fault orientations, namely, northwest (NW) and northeast (NE). For example, a dip of 60W does not belong to the plane with a particular fault with a given strike. Rather the data give the distribution of strike and dip independently.

A scheme has been devised here to express dip without any other information of direction, e.g., NW, NE, or W. As the faults are striking NE or NW, they will dip either to east or west. Therefore, in this scheme, the dip angle is always measured from the East direction. In other words, dip angle 0 degree and dipping towards east is dip 0. Similarly, dip angle 0 degree and dipping towards west is dip 180 degree. A dip angle of 30 degree and dipping east is dip 30 and dipping west is dip 120 degree.

It should be noted here that this scheme is not quite general and will not work if a fault has the strike towards east. In Yucca Mountain region, the fault orientation is either NE or NW. Therefore, this scheme will work without any problem. A better scheme is necessary to make this code more general and applicable to any other region.

[Amitava Ghosh, June 7, 1996]

2.1.8. Exponentially Distributed Random Numbers

Subroutine *exponential* generates a random sample x which is exponentially distributed with the parameter λ. The PDF of exponential distribution is

$$f(x;\lambda) = \lambda e^{-\lambda x} \qquad x \geq 0 \tag{A-33}$$

where λ is a constant. The mean and standard deviation of the exponential distribution are equal to $1/\lambda$.

The exponentially distributed random number, y, is generated using u, a uniformly distributed random number between 0 and 1 (Ang and Tang, 1984):

2.1.9. Determination of Waste Package Disruption

Subroutine *check_disrupt* calculates the consequence of a faulting event. It calculates the number and percentage of waste packages, and area of emplacement area disrupted for each faulting event. Inputs to this subroutine are the length and width of the fault, areal mass loading (MTU/acre), MTU per waste package, number of waste packages in the emplacement region, and the standoff distance. Standoff distance is the distance normal to the fault assumed to be affected by the movement along the fault plane.

2.1.10. SUBROUTINE *sort*

Subroutine *sort* has been adopted from Press et al. (1986). It sorts a real array of a given length into an ascending order using the Heapsort algorithm. The array is replaced by its sorted rearrangement. Details of the algorithm are available in Press et al. (1986)

2.2. Input Data

2.2.1. Simulation and Repository Regions

The FAULTING module requires information about the origin of the simulation area and the boundaries of different repository blocks which are given in terms of the coordinates of the four corners.

The origin of the simulation area was taken as 170500 m East and 233000 m North, which falls in the middle of the primary emplacement area. Different emplacement areas are given by (TRW Environmental Safety Systems Inc., 1995):

Region Primary Upper:
[(170500E, 234800N), (171300E, 234500N), (171100E, 231100N), (169800N, 231500E)]

Region Primary Lower:
[(171600E, 233700N), (172300E, 233300N), (171700E, 232000N), (171500E, 232100N)]

Optional Region A:
[(170800E, 236300N), (171300E, 236400N), (171950E, 235500N), (171300E, 235200N)]

Optional Region B:
[(169000E, 236800N), (169700E, 236500N), (169400E, 234500N), (168500E, 234800N)]

Optional Region C:
[(168300E, 234500N), (168900E, 234200N), (168200E, 232200N), (167600E, 232500N)]

Optional Region D:
[(169200E, 234200N), (169700E, 234000N), (169100E, 231500N), (168200E, 231700N)]

[Amitava Ghosh, March 31, 1996]

Figure 2-4 is from TRW Environmental Safety Systems Inc. (1995) showing the emplacement regions. Figure 2-5 shows the plots of the regions given by the above-listed coordinate points.

[Amitava Ghosh, June 7, 1996]

2.2.2. Development of Parameter File *fault.inc*

This code requires many site-specific values for different parameters. Previously, the values of these parameters were defined inside the code at the appropriate places where they were first defined. This may be helpful to read the code but has potential danger of accidentally modifying the parameters values during some modification of the code. Consequently, all these constants are declared in a separate file called *fault.inc* which is included in this code at the beginning of compilation. Any modification to the parameters can be done independently in the file *fault.inc* without any possibility of modifying any part of the code accidently. The content of the file *fault.inc* is given below:

```
c
c....C*****************************************************************
c     This file contains values of different variables used in the
c     FAULTING module. Any change to the data should be done here
c     so that the program remains unaffected.
c....C*****************************************************************
```

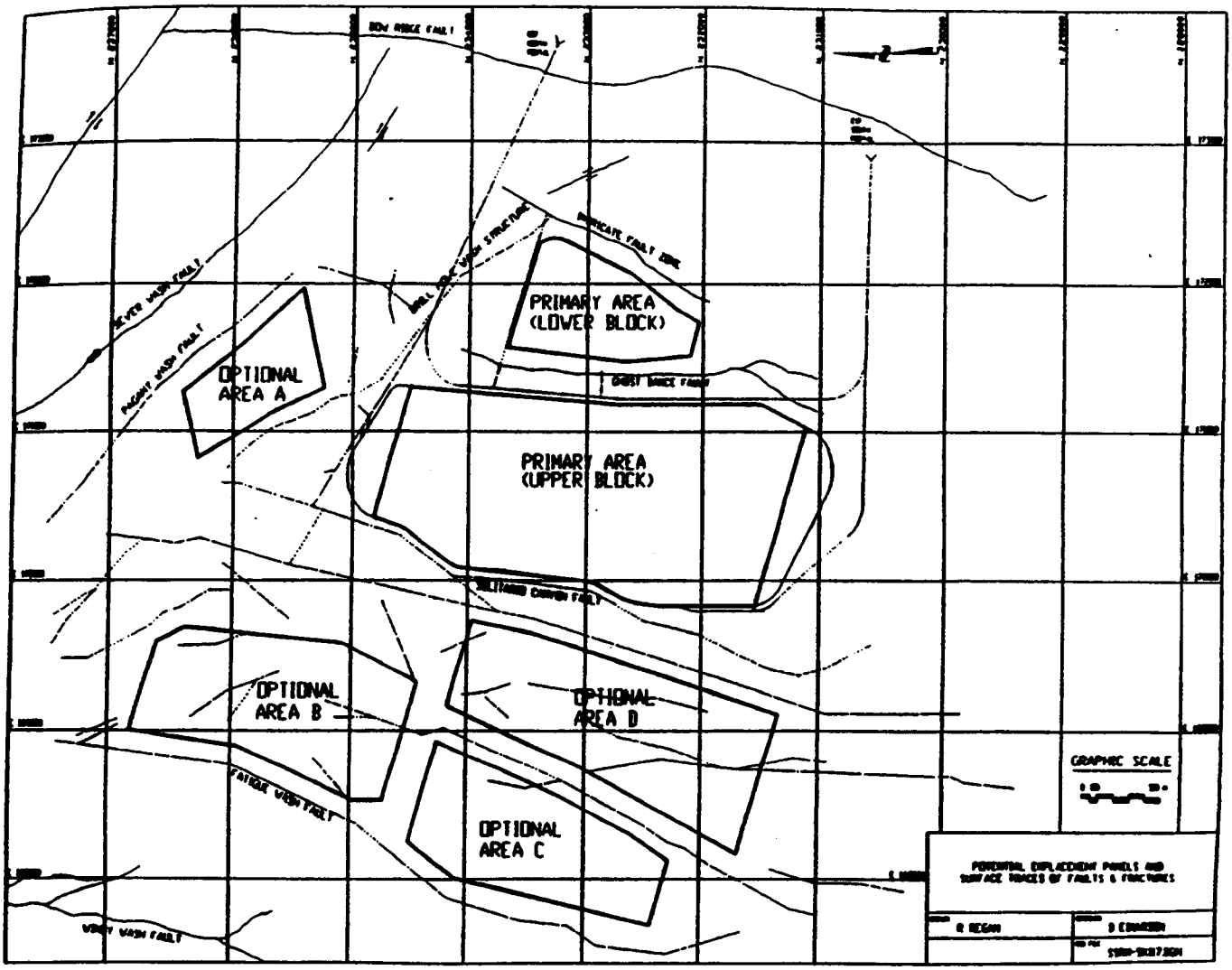

Figure 2-4 Different repository blocks (TRW Environmental Safety Systems Inc., 1995)

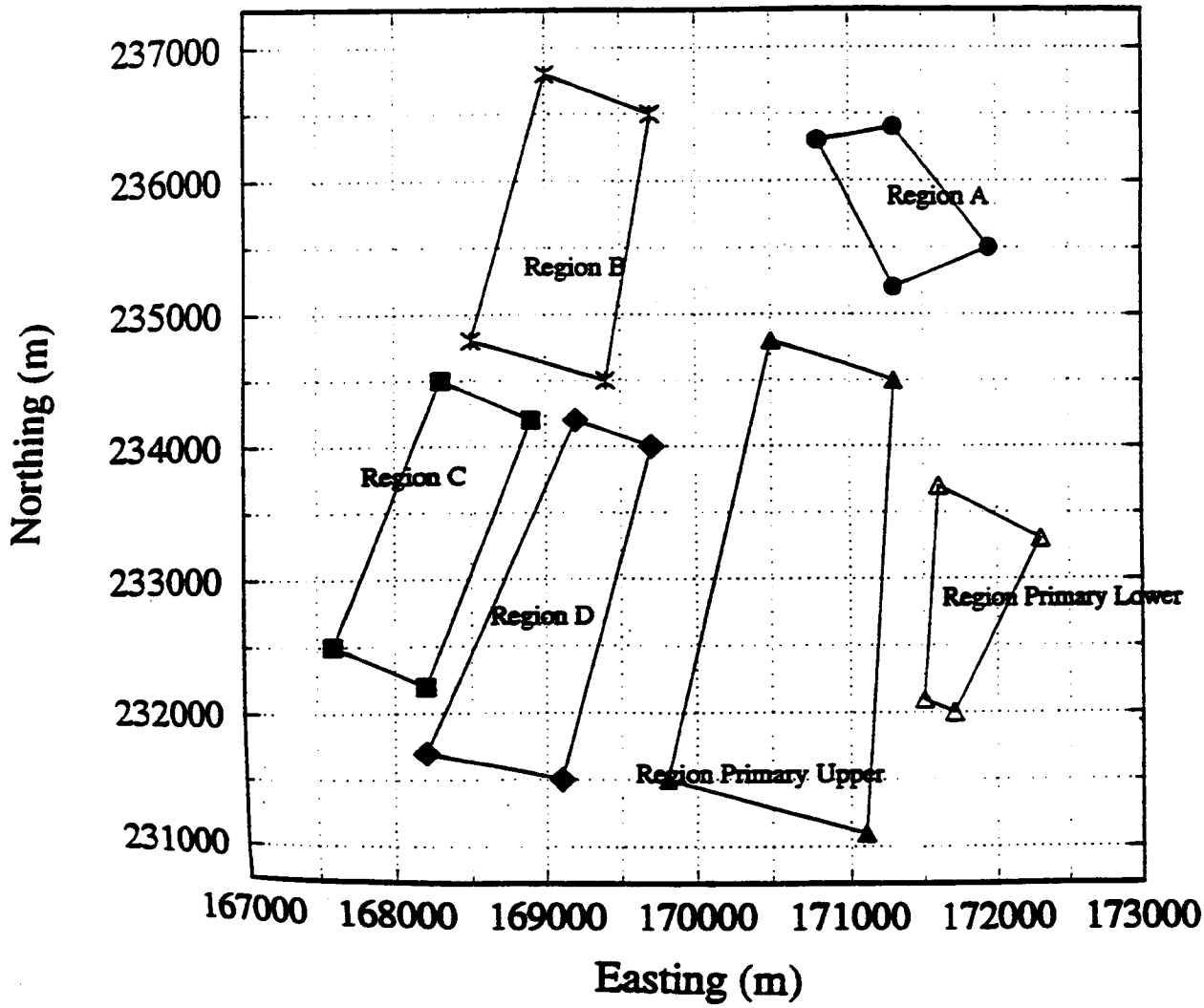

Figure 2-5 Different repository blocks as modeled in the FAULTING module

```
c
c
c... Coordinates of the square region (meter)
c
c
c     Define the coordinates of the square region
c     Origin is defined in the middle of the Primary Area (Upper Block)
c     x0 = 170500 East = 170500m, y0 = 233000 North = 233000 m
c     The simulation block is 50km x 50 km centered at (x0,y0)
c

      xleft   = 145500.0
      xright  = 195500.0
      ybottom = 208000.0
      ytop    = 258000.0

c
c... Probability of fault zone orientation
c

      pforient = 0.25

c
c... Strike of fault zone (degree, measured from North)
c

      min_SNW = 115.0d0
      max_SNW = 130.0d0

      min_SNE = 65.0d0
      max_SNE = 95.0d0

c
c... Total trace length (meter)
c

      min_ltNW = 2000.0d0
      max_ltNW = 10000.0d0

      min_ltNE = 3000.0d0
      max_ltNE = 12000.0d0
```

```
c
c... Dip angle (degree, measured from East)
c

      min_thetaNW = 80.0d0
      max_thetaNW = 100.0d0

      min_thetaNE = 90.0d0
      max_thetaNE = 120.0d0

c
c... Fault Width (meter)
c

      min_wNW = 0.5d0
      max_wNW = 275.0d0

      min_wNE = 0.5d0
      max_wNE = 365.0d0

      alpha = 1.5d0
      beta = 3.0d0

c
c... Time of coourance of largest credible displacement faulting event (year)
c

      min_t1 = 0.0d0
      max_t1 = 10000.0d0

c
c... Amount of largest credible displacement per faulting event (meter)
c

      min_LDNW = 4.5d-2
      max_LDNW = 25.0d-2

      min_LDNE = 6.0d-2
      max_LDNE = 45.0d-2

c
c... Largest credible displacement along multiple slip surfaces (meter)
```

```
c

      LD_max = 45.0d-2

c
c...  Amount of cumulative displacement rate (meter/year)
c

      min_CDNW = 0.0d-3
      max_CDNW = 0.01d-3

      min_CDNE = 0.00004d-3
      max_CDNE = 0.007d-3

c
c...  Threshold Displacement (meter)
c

      min_TD = 0.1d0
      max_TD = 0.5d0
```

[Amitava Ghosh, June 7, 1996]

Most of the variables in the include file has been converted to lower case. This was done to comply with the comment of D. Ferrill as SNW for strike for NW direction might be interpreted as South-North-West. New include file is given below:

```
c
c....C*****************************************************************
c       This file contains values of different variables used in the
c       FAULTING module. Any change to the data should be done here
c       so that the program remains unaffected.
c....C*****************************************************************
c

c
c... Coordinates of the square region (meter)
c
c
c       Define the coordinates of the square region
c       Origin is defined in the middle of the Primary Area (Upper Block)
c       x0 = 170500 East = 170500m, y0 = 233000 North = 233000 m
c       The simulation block is 50 km x 50 km centered at (x0,y0)
c

        xleft   = 145500.0
        xright  = 195500.0
```

```
        ybottom = 208000.0
        ytop    = 258000.0

c
c...  Probability of fault zone orientation
c

        pforient = 0.25

c
c...  Strike of fault zone (degree, measured from North)
c

        min_sNW = 115.0d0
        max_sNW = 130.0d0

        min_sNE = 65.0d0
        max_sNE = 95.0d0

c
c...  Total trace length (meter)
c

        min_ltNW = 2000.0d0
        max_ltNW = 10000.0d0

        min_ltNE = 3000.0d0
        max_ltNE = 12000.0d0

c
c...  Dip angle (degree, measured from East)
c

        min_thetaNW = 80.0d0
        max_thetaNW = 100.0d0

        min_thetaNE = 90.0d0
        max_thetaNE = 120.0d0

c
c...  Fault Width (meter)
c

        min_wNW = 0.5d0
        max_wNW = 275.0d0

        min_wNE = 0.5d0
        max_wNE = 365.0d0

        alpha = 1.5d0
        beta = 3.0d0

c
c...  Time of occurrence of largest credible displacement faulting event
(year)
```

```
c     Changed from 0 and 10,000 years to 60,000 and 275,000 years
c
       min_t1 = 60000.0d0
       max_t1 = 275000.0d0

c
c...  Amount of largest credible displacement per faulting event (meter)
c
       min_ldNW = 4.5d-2
       max_ldNW = 25.0d-2

       min_ldNE = 6.0d-2
       max_ldNE = 45.0d-2

c
c...  Largest credible displacement along multiple slip surfaces (meter)
c
       ld_max = 45.0d-2

c
c...  Amount of cumulative displacement rate (meter/year)
c
       min_cdNW = 0.0d-3
       max_cdNW = 0.01d-3

       min_cdNE = 0.00004d-3
       max_cdNE = 0.007d-3

c
c...  Define standoff distance
c
       standoff = 0.0

c
c...  Define MTU per waste package
c
       MTU_wp = 9.0
```

2.2.3. Function *drand_a()*

A new function *drand_a()* is inserted in the code. Function *drand()* of the SUN compiler has been used to generate uniformly distributed random numbers. To make the code portable, function *drand_a()* is inserted as an alternative to the *darnd()* function. This function has been developed from Ripley (1987). Everywhere function *drand()* has been called, the line has to be replaced by the call to *drand_a()*. This function does not take any argument. The seed for random number generation comes through the COMMON statement. Moreover, both *seed* and *drand_a* have to be defined as REAL*8 in each routine

where *drand_a()* will be used. These codes are all there in the code. They have to be uncommented and calls to *darand()* function have to be commented.

3. VERIFICATION OF DIFFERENT PARTS OF THE FAULTING MODULE

3.1. Calculation of Intersected Length of a Fault

In this case, the coordinates of two ends of four faults were tested. The faults are shown in Figure 3-1. Four faults were tested: Fault 1 and Fault 2 fall completely within the repository region, in this case Region A. One end of Fault 3 intersects the region (only one end inside the region). Both ends of Fault 4 are outside the region, as shown in Figure 3-1. The coordinates of the both ends of the faults are given below along with the fault length and the length of the fault intersected by the repository region A.

Fault 1:
End 1: [171200, 236200] End 2: [171400, 235600]
Length = 632.456
Intersected Length = 632.456

Fault 2:
End 1: [171300, 235600] End 2: [171300, 235300]
Length = 300.000
Intersected Length = 300.000

Fault 3:
End 1: [170900, 235400] End 2: [171000, 236000]
Length = 608.276
Intersected Length = 103.852

Fault 4:
End 1: [171500, 235200] End 2: [171750, 236000]
Length = 838.153
Intersected Length = 562.013

Figures 3-2 and 3-3 show Fault 3 and Fault 4 and the segments of the region intersected (real intersection) by them. Following are the results from the FAULTING module which shows good agreements with the results.

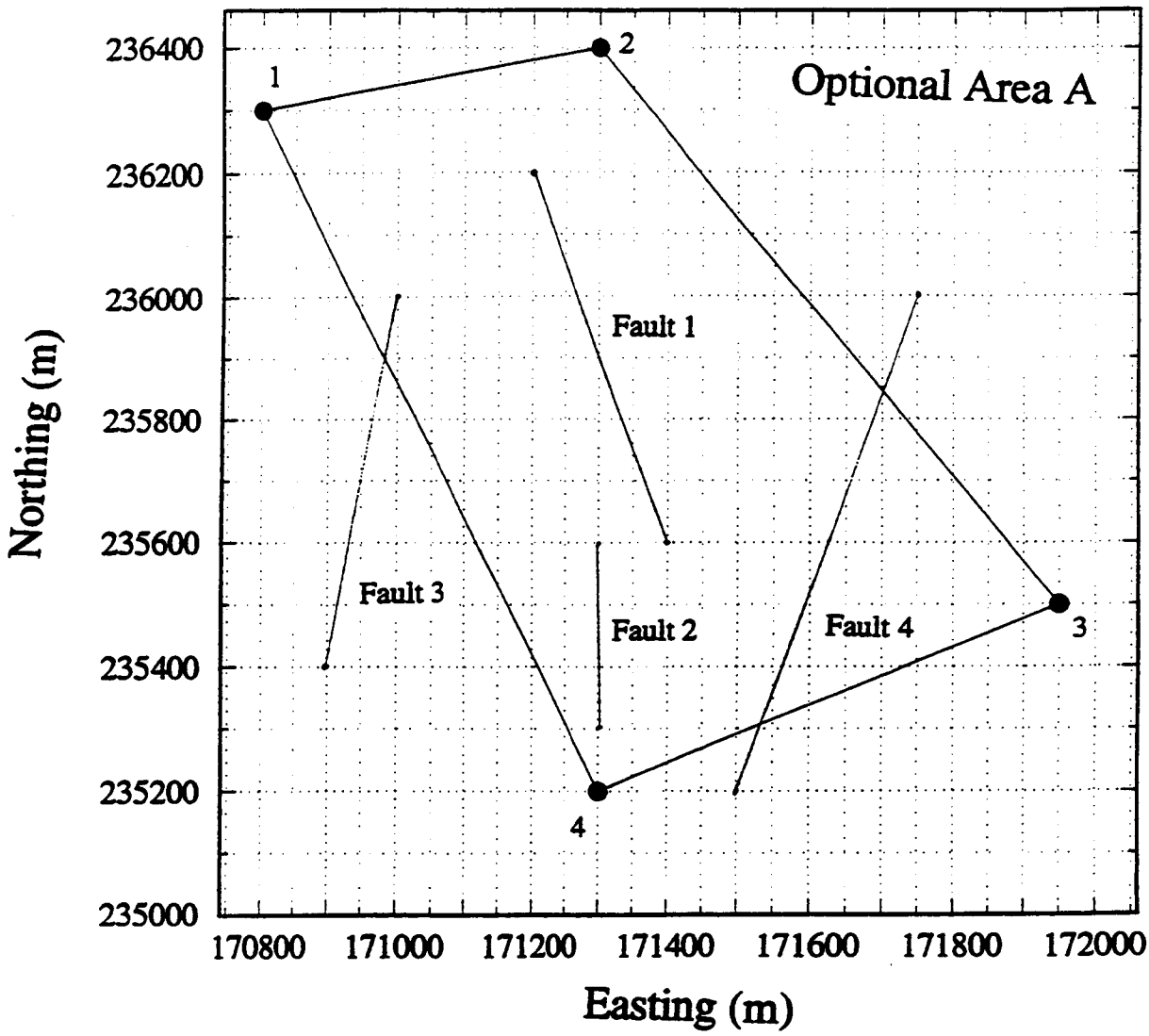

Figure 3-1 Region A with four fault planes used in verification of the code

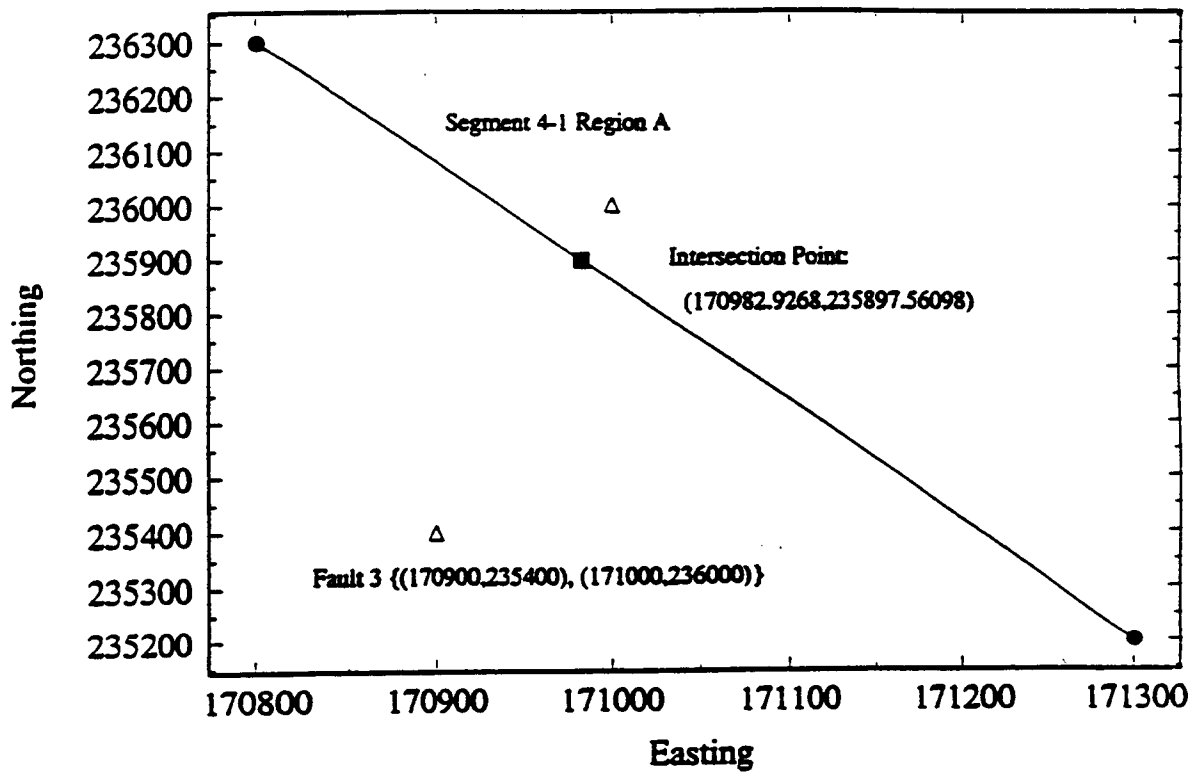

Figure 3-2 Intersection point of Fault 3 and segments of Region A

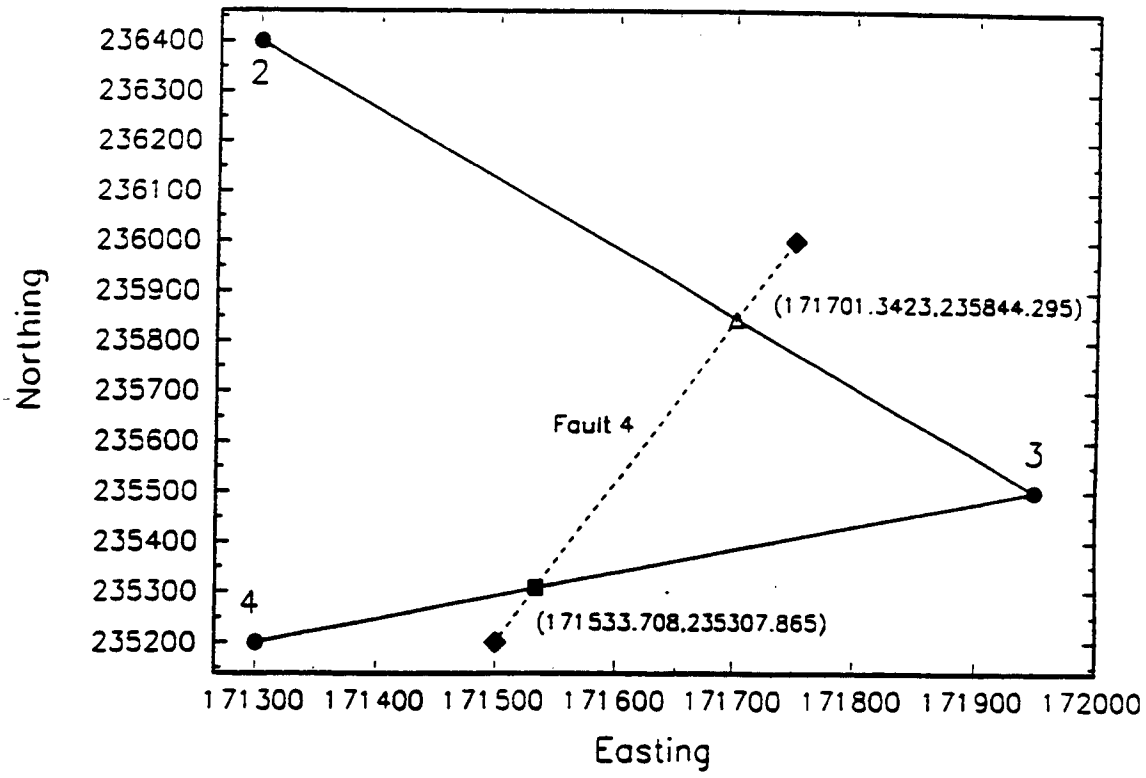

Figure 3-3 Intersection points of Fault 4 and segments of Region A

FAULT 1:

x1 of fault => 171200
y1 of fault => 236200
x2 of fault => 171400
y2 of fault => 235600
Region (PU,PL,A,B,C,D) => A

... Fault Plane ...
End 1: x = 171200.00000000 y = 236200.00000000
End 2: x = 171400.00000000 y = 235600.00000000
Region: A

x(1) = 170800.0 y(1) = 236300.0
x(2) = 171300.0 y(2) = 236400.0
x(3) = 171950.0 y(3) = 235500.0
x(4) = 171300.0 y(4) = 235200.0
x(5) = 170800.0 y(5) = 236300.0

171200.00000000 236200.00000000 falls inside the region
171400.00000000 235600.00000000 falls inside the region

Intersection Points:
Point 1: 171200.00000000 236200.00000000
Point 2: 171400.00000000 235600.00000000
Intersected length = 632.45553203368

Another run => y

FAULT 2:

x1 of fault => 171300
y1 of fault => 235600
x2 of fault => 171300
y2 of fault => 235300
Region (PU,PL,A,B,C,D) => A

... Fault Plane ...
End 1: x = 171300.00000000 y = 235600.00000000
End 2: x = 171300.00000000 y = 235300.00000000
Region: A

x(1) = 170800.0 y(1) = 236300.0
x(2) = 171300.0 y(2) = 236400.0
x(3) = 171950.0 y(3) = 235500.0

x(4) = 171300.0 y(4) = 235200.0
x(5) = 170800.0 y(5) = 236300.0

171300.00000000 235600.00000000 falls inside the region
171300.00000000 235300.00000000 falls inside the region

Intersection Points:
Point 1: 171300.00000000 235600.00000000
Point 2: 171300.00000000 235300.00000000
Intersected length = 300.00000000000

Another run => y

FAULT 3:

x1 of fault => 170900
y1 of fault => 235400
x2 of fault => 171000
y2 of fault => 236000
Region (PU,PL,A,B,C,D) => A

... Fault Plane ...
End 1: x = 170900.00000000 y = 235400.00000000
End 2: x = 171000.00000000 y = 236000.00000000
Region: A

x(1) = 170800.0 y(1) = 236300.0
x(2) = 171300.0 y(2) = 236400.0
x(3) = 171950.0 y(3) = 235500.0
x(4) = 171300.0 y(4) = 235200.0
x(5) = 170800.0 y(5) = 236300.0

170900.00000000 235400.00000000 falls outside the region
171000.00000000 236000.00000000 falls inside the region

In subroutine Intersection
x1 = 170800.00000000 y1 = 236300.00000000
x2 = 171300.00000000 y2 = 236400.00000000
x3 = 170900.00000000 y3 = 235400.00000000
x4 = 171000.00000000 y4 = 236000.00000000
Part 1 = -16078881000000. Part 2 = 16128488000000.
Part 4 = -27817218000000. Part 5 = 27885760000000.
Part 6 = -10000.0000000000 Part 7 = 300000.00000000
xi = 171058.62068966 yi = 236351.72413793

Subroutine Inlength

Data about Intersection:
Segment: 1
x1 = 170800.00000000 y1 = 236300.00000000
x2 = 171300.00000000 y2 = 236400.00000000
Intersection Point:
xi = 171058.62068966 yi = 236351.72413793
A = 356.57573453473 B = 964.85198756455 C = 246.15956272518
D = 263.74238863410 E = 509.90195135928 F = 608.27625302982

In subroutine Intersection
x1 = 171300.00000000 y1 = 236400.00000000
x2 = 171950.00000000 y2 = 235500.00000000
x3 = 170900.00000000 y3 = 235400.00000000
x4 = 171000.00000000 y4 = 236000.00000000
Part 1 = -22099812000000. Part 2 = 22181945000000.
Part 4 = -36235122000000. Part 5 = 36348720000000.
Part 6 = 90000.000000000 Part 7 = 390000.00000000
xi = 171110.41666667 yi = 236662.50000000

Subroutine Inlength
Data about Intersection:
Segment: 2
x1 = 171300.00000000 y1 = 236400.00000000
x2 = 171950.00000000 y2 = 235500.00000000
Intersection Point:
xi = 171110.41666667 yi = 236662.50000000
A = 671.63836272043 B = 1279.9146157502 C = 1433.9827138467
D = 323.80254828797 E = 1110.1801655587 F = 608.27625302982

In subroutine Intersection
x1 = 171950.00000000 y1 = 235500.00000000
x2 = 171300.00000000 y2 = 235200.00000000
x3 = 170900.00000000 y3 = 235400.00000000
x4 = 171000.00000000 y4 = 236000.00000000
Part 1 = 30198825000000. Part 2 = -30260324000000.
Part 4 = 35966766000000. Part 5 = -36051360000000.
Part 6 = 30000.000000000 Part 7 = -390000.00000000
xi = 170830.55555556 yi = 234983.33333333

Subroutine Inlength
Data about Intersection:
Segment: 3
x1 = 171950.00000000 y1 = 235500.00000000
x2 = 171300.00000000 y2 = 235200.00000000
Intersection Point:
xi = 170830.55555556 yi = 234983.33333333

A = 1030.6903176339 B = 422.41406460403 C = 517.03242728497
D = 1232.9234804488 E = 715.89105316382 F = 608.27625302982

In subroutine Intersection
x1 = 171300.00000000 y1 = 235200.00000000
x2 = 170800.00000000 y2 = 236300.00000000
x3 = 170900.00000000 y3 = 235400.00000000
x4 = 171000.00000000 y4 = 236000.00000000
Part 1 = 24143916000000. Part 2 = -24214019000000.
Part 4 = 27760722000000. Part 5 = -27857440000000.
Part 6 = -110000.00000000 Part 7 = -300000.00000000
xi = 170982.92682927 yi = 235897.56097561

Subroutine Inlength
Data about Intersection:
Segment: 4
x1 = 171300.00000000 y1 = 235200.00000000
x2 = 170800.00000000 y2 = 236300.00000000
Intersection Point:
xi = 170982.92682927 yi = 235897.56097561
A = 103.852043200217 B = 504.42420982961 C = 442.06265757053
D = 766.24193978893 E = 1208.3045973595 F = 608.27625302982
Intersection is real

k = 2 xin(k) = 170982.92682927 yin(k) = 235897.56097561

Intersection Points (In subroutine segment):
Point 1: 171000.00000000 236000.00000000
Point 2: 170982.92682927 235897.56097561
Intersection Points:
Point 1: 171000.00000000 236000.00000000
Point 2: 170982.92682927 235897.56097561

Intersected length = 103.852043200217

Another run => y

FAULT 4:
x1 of fault => 171500
y1 of fault => 235200
x2 of fault => 171750
y2 of fault => 236000
Region (PU,PL,A,B,C,D) => A

... Fault Plane ...

End 1: x = 171500.00000000 y = 235200.00000000
End 2: x = 171750.00000000 y = 236000.00000000
Region: A

x(1) = 170800.0 y(1) = 236300.0
x(2) = 171300.0 y(2) = 236400.0
x(3) = 171950.0 y(3) = 235500.0
x(4) = 171300.0 y(4) = 235200.0
x(5) = 170800.0 y(5) = 236300.0

171500.00000000 235200.00000000 falls outside the region
171750.00000000 236000.00000000 falls outside the region

In subroutine Intersection
x1 = 170800.00000000 y1 = 236300.00000000
x2 = 171300.00000000 y2 = 236400.00000000
x3 = 171500.00000000 y3 = 235200.00000000
x4 = 171750.00000000 y4 = 236000.00000000
Part 1 = -10078252500000.0 Part 2 = 10142720000000.0
Part 4 = -27811224000000. Part 5 = 27899920000000.
Part 6 = -25000.000000000 Part 7 = 400000.00000000
xi = 171913.33333333 yi = 236522.66666667

Subroutine Inlength
Data about Intersection:
Segment: 1
x1 = 170800.00000000 y1 = 236300.00000000
x2 = 171300.00000000 y2 = 236400.00000000
Intersection Point:
xi = 171913.33333333 yi = 236522.66666667
A = 547.59311739851 B = 1385.7458481105 C = 625.47972700072
D = 1135.3816783600 E = 509.90195135928 F = 838.15273071201

In subroutine Intersection
x1 = 171300.00000000 y1 = 236400.00000000
x2 = 171950.00000000 y2 = 235500.00000000
x3 = 171500.00000000 y3 = 235200.00000000
x4 = 171750.00000000 y4 = 236000.00000000
Part 1 = -16094895000000. Part 2 = 16222812500000.
Part 4 = -36045576000000. Part 5 = 36221280000000.
Part 6 = 225000.00000000 Part 7 = 520000.00000000
xi = 171701.34228188 yi = 235844.29530201

Subroutine Inlength
Data about Intersection:
Segment: 2

x1 = 171300.00000000 y1 = 236400.00000000
x2 = 171950.00000000 y2 = 235500.00000000
Intersection Point:
xi = 171701.34228188 yi = 235844.29530201
A = 163.13039725267 B = 675.02233345934 C = 424.69979487818
D = 685.48037068055 E = 1110.1801655587 F = 838.15273071201
Intersection is real

k = 1 xin(k) = 171701.34228188 yin(k) = 235844.29530201

In subroutine Intersection
x1 = 171950.00000000 y1 = 235500.00000000
x2 = 171300.00000000 y2 = 235200.00000000
x3 = 171500.00000000 y3 = 235200.00000000
x4 = 171750.00000000 y4 = 236000.00000000
Part 1 = 36342427500000. Part 2 = -36418760000000.
Part 4 = 35989128000000. Part 5 = -36093840000000.
Part 6 = 75000.000000000 Part 7 = -520000.00000000
xi = 171533.70786517 yi = 235307.86516854

Subroutine Inlength
Data about Intersection:
Segment: 3
x1 = 171950.00000000 y1 = 235500.00000000
x2 = 171300.00000000 y2 = 235200.00000000
Intersection Point:
xi = 171533.70786517 yi = 235307.86516854
A = 725.14337376209 B = 113.00935694992 C = 257.39903035103
D = 458.49202281278 E = 715.89105316382 F = 838.15273071201
Intersection is real

k = 2 xin(k) = 171533.70786517 yin(k) = 235307.86516854

In subroutine Intersection
x1 = 171300.00000000 y1 = 235200.00000000
x2 = 170800.00000000 y2 = 236300.00000000
x3 = 171500.00000000 y3 = 235200.00000000
x4 = 171750.00000000 y4 = 236000.00000000
Part 1 = 30240840000000. Part 2 = -30356547500000.
Part 4 = 27543096000000. Part 5 = -27701680000000.
Part 6 = -275000.00000000 Part 7 = -400000.00000000
xi = 171418.51851852 yi = 234939.25925926

Subroutine Inlength
Data about Intersection:
Segment: 4

```
x1 =    171300.00000000 y1 =    235200.00000000
x2 =    170800.00000000 y2 =    236300.00000000
Intersection Point:
xi =    171418.51851852 yi =    234939.25925926
A =     1111.3284355367 B =     273.17570482465 C =     1494.7175389558
D =     286.41294159630 E =     1208.3045973595 F =     838.15273071201
Intersection Points:
Point 1:    171701.34228188    235844.29530201
Point 2:    171533.70786517    235307.86516854

Intersected length =    562.01297650942


                    Another run => n
```

[Amitava Ghosh, June 7, 1996]

3.2. Simulation of Uniformly Distributed Random Numbers Between [0,1]

Uniformly distributed random numbers between 0 and 1 are simulated using the *drand(k)* function of SUNf77 compiler where k>0 starts a new sequence of random numbers and k = 0 gives the next random number in the sequence. *drand(k)* function gives the double precision random numbers.

Using these random numbers, a set of random numbers uniformly distributed between [a,b] can be generated from

$$y = (b - a)\, x + a$$

where,
x = uniform random number between [0,1]
y = uniform random number between [a,b].

A subroutine to generate uniformly distributed random numbers between [0,1], called *random*, is left in the file *fault.f* although it was not used to generate the uniformly distributed random numbers. This subroutine may be used in place of the *drand(k)* function with some modifications of the procedure for calling for the random number in *fault.f* as this subroutine requires the previous random number as seed for generation for next random number.

[Amitava Ghosh, June 27, 1996]

3.3. Simulation of Normally Distributed Random Numbers

A subroutine *gauss* has been developed following the mathematical guideline given in Section 2.1.4 and adopting the routine given in Ghosh and Kulatilake (1987). A set of 500 normally distributed random numbers with a probability 90% falling between [-1, 1] was generated. The statistics of the generated numbers are given below:

	Input Data	**Normal(3.74e-3,0.59)**
Minimum	-1.584767	
Maximum	1.966892	
Mode	-0.128587	3.743165e-3
Mean	3.743165e-3 .	3.743165e-3
Std Deviation	0.586337	0.586337
Variance	0.343791	0.343791
Skewness	0.119546	0.0
Kurtosis	3.05641	3.0

Best Fit Results	
C-S Test	57.136959
K-S Test	0.023695
A-D Test	0.231722

Figure 3-4 shows the histogram of random numbers with a fitted normal distribution with mean and standard deviation equal to 0.0037 and 0.5863 respectively. They compare very well with the population mean and standard deviation equal to 0.0 and 0.6079 respectively. The skewness and kurtosis of the data set indicate normality. Figure 3-5 shows the cumulative distribution of these numbers. The numbers were tested for normality. The data set passes Kolmogorov-Smirnov (K-S) test for goodness-of-fit with $\alpha = 0.05$. As expected, with a larger set of random numbers, the sample mean and standard deviation become closer to the population mean and standard deviation. This has been tested with different data sets. For example, when the number of generated random numbers is 1000, the sample mean and standard deviation are 0.0012 and 0.5987 respectively, as shown below. The skewness and kurtosis of the data set have improved quite a bit.

	Input Data	**Normal(1.25e-3,0.60)**
Minimum	-2.051897	
Maximum	1.859709	
Mode	-0.056978	1.249021e-3
Mean	1.249021e-3	1.249021e-3
Std Deviation	0.598718	0.598718
Variance	0.358463	0.358463

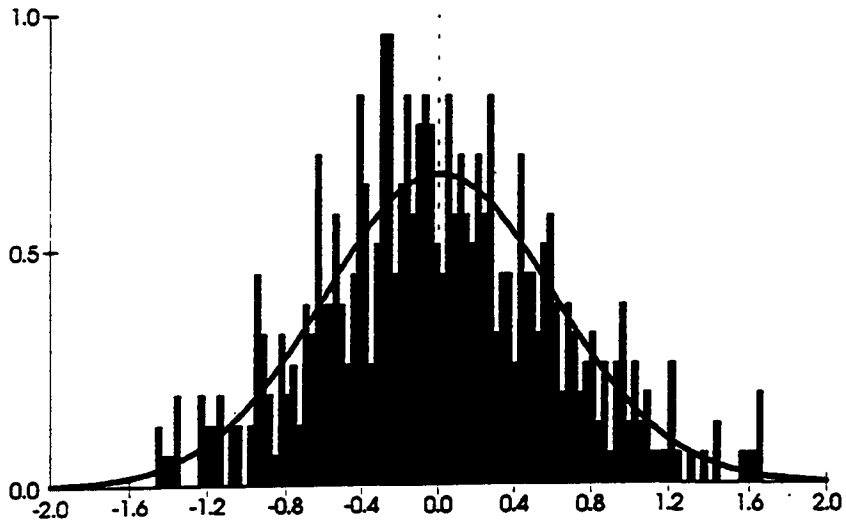

Figure 3-4 Histogram of random numbers, generated by *gauss*, having normal distribution

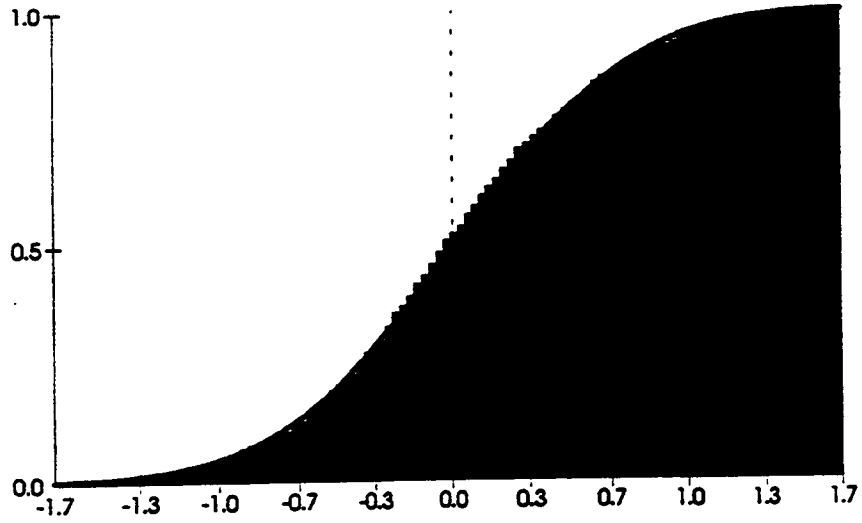

Figure 3-5 Cumulative distribution of random numbers generated by *gauss* having normal distribution

Skewness	-0.012203	0.0
Kurtosis	2.969442	3.0

Best Fit Results

C-S Test	37.568098
K-S Test	8.731377e-3
A-D Test	0.077632

[Amitava Ghosh, June 28, 1996]

3.4. Simulation of Beta Distributed Random Numbers

A subroutine *betad* has been developed following the mathematical guideline given in Section 2.1.5 with any acceptable values of α and β. In the file *fault.inc*, the values of α and β are fixed as 1.5 and 3.0, as given in Stirewalt et al. (1995). A set of 1000 random numbers between [0, 1] was generated. Sample statistics show that sample α and β are 1.50 and 3.11 respectively. Sample mean and variance are 0.3279 and 0.0390 respectively which compare very well with population mean of 0.3333 and population variance of 0.0404. Figure 3-6 shows the histogram of random numbers with beat distribution fitted over it.

	Input Data	**Beta(1.50,3.11) + 2.92e-3**
Minimum	2.91705e-3	
Maximum	0.92994	
Mode	0.206862	0.191073
Mean	0.327924	0.325007
Std Deviation	0.197694	0.197694
Variance	0.039083	0.039083
Skewness	0.485402	0.535402
Kurtosis	2.421904	3.38095

Best Fit Results

C-S Test	13.767742
K-S Test	0.013351
A-D Test	0.258163

[Amitava Ghosh, June 28, 1996]

3.5. Verification of Subroutine *exponential*

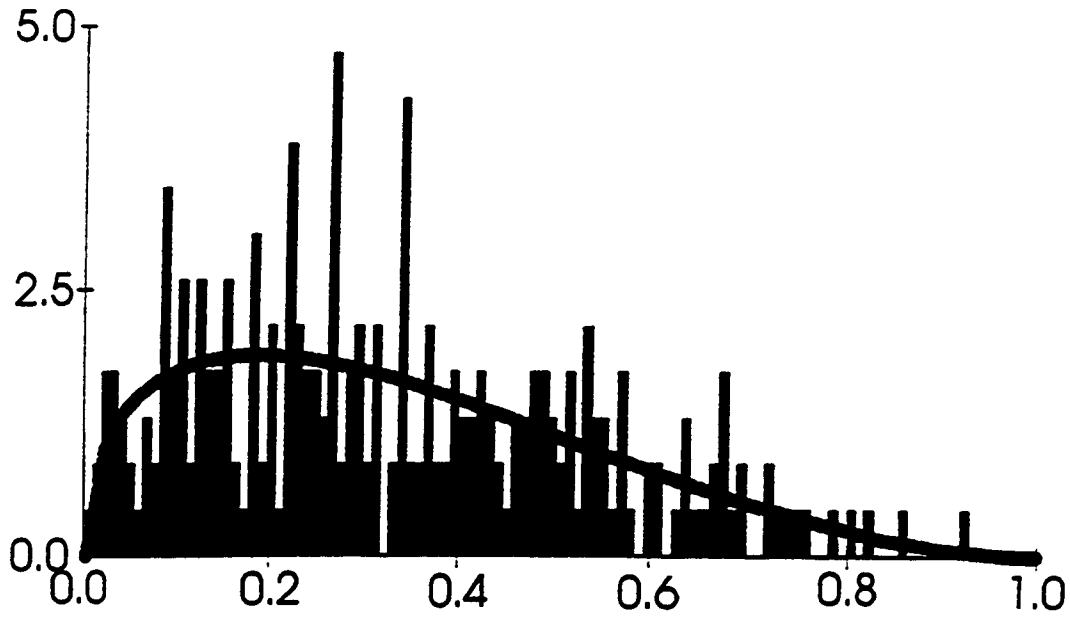

Figure 3-6 Histogram of random numbers generated by *betad* having beta distribution

Random numbers following the exponential distribution with the parameter λ are simulated using the subroutine *exponential*. The equation used in the subroutine *exponential* is given in Section A.10. A set of 1000 random numbers with λ equal to 10 (mean and standard deviation equal to 0.1) were generated. The statistics of the generated numbers are given below:

	Input Data	Exponential Distribution
Minimum	1.85602e-4	
Maximum	0.62722	
Mode	9.59108e-3	
Mean	0.09888	0.09888
Standard Deviation	0.095711	0.09888
Variance	9.16051e-3	9.77780e-3
Skewness	1.75468	2.0
Kurtosis	6.97331	9.0

Best Fit Results

Chi-Square Test	113.74484
Kolmogorov-Smirnov Test	0.02051
Anderson-Darling Test	0.31278

Figure 3-7 shows the histogram of generated random numbers with a fitted exponential distribution. The mean and standard deviation compare very well with the given λ. Goodness-of-fit analysis with Chi-Square, Kolmogorov-Smirnov, and Anderson-Darling tests show that the data set is exponential with α equal to 0.05.

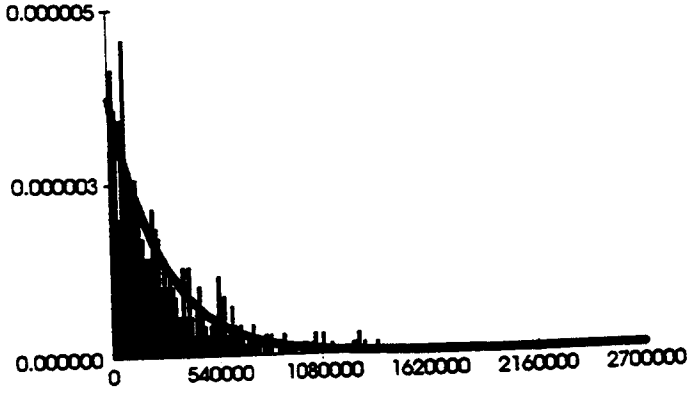

Figure 3-7. Histogram of random numbers generated by exponential distribution

[Amitava Ghosh, December 15, 1996]

3.6. Verification of Subroutine *check_disrupt*

Subroutine *check_disrupt* has been verified by developing a small test program which basically supplies the values of all input variables of the subroutine and prints the output on the screen. These results were checked using hand-calculation.

A typical output for verification run of the subroutine *check_disrupt* is given below. For this particular run, a fault having length of 1000 m within the given repository was assumed. The fault had a width of 10 m. Each waste package contained 10 MTU of waste. Waste packages were emplaced at an AML of 100 MTU/acre. There is no standoff distance. The area affected should be equal to 10000 m^2 or 2.47097 acre. Number of waste packages disrupted should be equal to 25 or 0.36 percent of all waste packages.

```
                    length => 1000

                     width => 10

                  standoff => 0

                       AML => 100

                    MTU_wp => 10
 Number of waste packages disrupted = 25
 Percent of waste packages disrupted = 0.35709184408188 percent
 Area disrupted =     2.4709700000000 acre
```

[Amitava Ghosh, December 15, 1996]

3.7. Verification of Subroutine *sort*

Subroutine *sort* of Press et al. (1986) has been verified using a small test program which takes the values of all elements in an array and calls the subroutine *sort* to rearrange them in an ascending order. It then prints the rearranged values on the screen. A typical example of the verification run is given below.

```
                    number of data => 20
 Data  1 => 1
 Data  2 => 4
```

```
 Data  3 => 5
 Data  4 => 6
 Data  5 => 7
 Data  6 => 8
 Data  7 => 2
 Data  8 => 3
 Data  9 => 20
 Data 10 => 19
 Data 11 => 18
 Data 12 => 10
 Data 13 => 11
 Data 14 => 12
 Data 15 => 17
 Data 16 => 16
 Data 17 => 13
 Data 18 => 15
 Data 19 => 14
 Data 20 => 0
   0.
     1.0000000000000
     2.0000000000000
     3.0000000000000
     4.0000000000000
     5.0000000000000
     6.0000000000000
     7.0000000000000
     8.0000000000000
    10.0000000000000
     11.000000000000
     12.000000000000
     13.000000000000
     14.000000000000
     15.000000000000
     16.000000000000
     17.000000000000
     18.000000000000
     19.000000000000
     20.000000000000
```

[Amitava Ghosh, December 15, 1996]

3.8. Verification of Function *drand_a()*

Function drand_a() was verified using a small test program which gives a seed to drand_a() and generated a specified number of uniformly distributed sample by calling the function drand_a(). A typical example of the verification run is given below. A total of 5000 random samples were generated. Only first 50 are show below for clarity. The histogram of all 5000 samples are shown in Figure 3-8.

0.31846297128054
0.40715831211170
0.10975166135922
0.59617246435777
0.87060846102918
0.31640451742169
0.81072430629783
0.84341594755809
0.29183060875713
0.79704138114911
0.87449297303077
0.60339772822494
0.30561827649624
0.52637307230680
0.75222626037534
0.66675812828669
0.20386211443872
0.31055737161569
0.53774474493123
0.87592805916254
0.72289034478501
0.61802480165755
0.14284145838713
0.73639111255128
0.52542864928275
0.87930849514869
0.53787796410633
0.11494273511457
0.84254907064259
0.72223028993338
0.52448291029990
0.98427341039492
0.68320850733817
0.68538283262652
0.22926795400179
0.30650290814531
0.39437719825393
0.29757105386703
0.27670234314944
0.53628131259991
0.28002086667345
0.31070618066504

3.8778437319574D-02
0.74919603008274
0.73767760057825
0.14743291872900
0.90506507824411
0.42877004874347
0.33820923154159
0.28255451949432

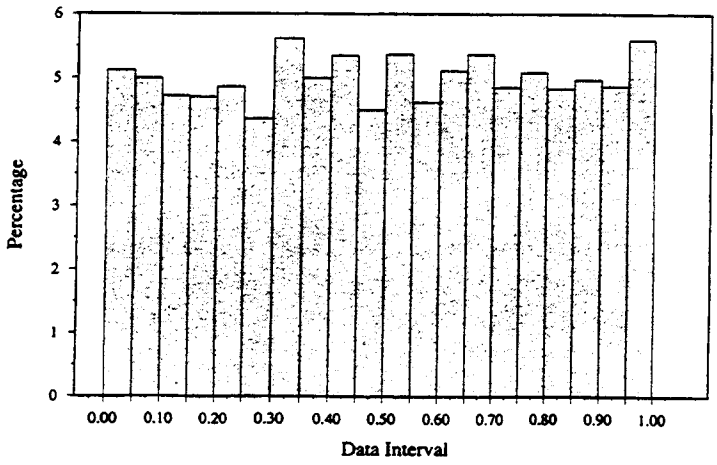

Figure 3.8. Histogram of 5000 random number generated using drand_a function

4. OTHER FEATURES

4.1. Date and Time Stamp

This code will write the date and time of running on the output file. It uses the *fdate()* function from the SUNf77 library.

[Amitava Ghosh, June 27, 1996]

4.2. File for Plotting the Faults and the Repository Region

At present, CNWRA does not have any software which can plot the faults and the repository while the module is running. Only available software is a freeware called PLOTMTV which does a decent

header information for formatting and manipulating the visual look of the plot. The FAULTING module automatically writes the necessary information to the plotting files for processing by PLOTMTV. In most cases, the plot file will run without any modification on PLOTMTV and produce a decent plot. User may want to change the plot file for enhancement of the plot. PLOTMTV can produce a postscript output for plotting on postscript printer. Present setup in the FAULTING module only produces black and white output. With trivial modification and proper setting of PLOTMTV program, a color output can be generated.

[Amitava Ghosh, June 27, 1996]

4.3. Development of Check for Emplacement Area Requirement

At the beginning of the simulation, the code checks whether the emplacement area available in Primary Upper region is sufficient to dispose 70,000 MTU of waste at the Areal Mass Loading (AML) (MTU/acre) specified before. If the available area is not sufficient, next emplacement region check is primary Lower. If the area is still not sufficient, the Optional Area A, Optional Area B, Option Area C, and Optional Area D will be checked in this sequence. The program keeps track of all the emplacement regions necessary to run a specific case. The program checks all the necessary emplacement regions and sums the length of the fault in each region to calculate the length of a fault within the repository.

[Amitava Ghosh, December 20, 1996]

4.4. Development of User's Manual

A user's manual for the FAULTING module has been prepared. It is named "FAULTIMG Version 1.0 – A Code for Simulation of Direct Fault Disruption Technical Description and User's Guide" (CNWRA 97-002), authored by A. Ghosh, R.D. Manteufel, and G. Stirewalt. The report documents the technical details, mathematical theories, details of compilation of the code, and required input and output of the code. Results of verification of the subroutines used in the code are also given.

[Amitava Ghosh, January 30, 1997]

4.5. Quality Assurance Status

The FAULTING module is not currently under TOP-018 but will be placed under control when Total-system Performance Assessment version 3 (TPA 3) code is placed under TOP-018.

[Amitava Ghosh, February 12, 1996]

5. REFERENCES

Ang, A.H-S., and W.H. Tang. 1984. *Probability Concepts in Engineering Planning and Design Volume II: Decision, Risk, and Reliability*. New York, NY: John Wiley & Sons.

Iman, R.L., and M.J. Shortencarier. 1984. *A FORTRAN 77 Program and User's Guide for the Generation of Latin Hypercube and Random Samples for Use With Computer Models*. SAND83-2365 and NUREG/CR-3624. Albuquerque, NM: Sandia National Laboratories.

Stirewalt, G.L., S.M. McDuffie, R.D. Manteufel, and R.W. Janetzke. 1995. *Technical Specifications for a Fault Displacement Module*. Report to Nuclear Regulatory Commission. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.

TRW Environmental Safety Systems Inc. 1995. *Total System Performance Assessment – 1995: An Evaluation of the Potential Yucca Mountain Repository*. Report to U.S. Department of Energy, Office of Civilian Radioactive Waste Management, Las Vegas, NV: TRW Environmental Safety Systems Inc.

Ghosh, A. and P.H.S.W. Kulatilake. 1987. A FORTRAN Program for Generation of Multivariate Normally Distributed Random Numbers. *Computers & Geosciences* 13(3): 221-233.

[Amitava Ghosh, June 28, 1996]

Press, W.H., B.P. Flannery, S.A. Tekolsky, and W.T. Vetterling. 1986. *Numerical Recipes in FORTRAN*. Cambridge, U.K.: Cambridge University Press.

[Amitava Ghosh, December 20, 1996]

Ripley, B.D. 1987. *Stochastic Simulation*. New York, NY: John Wiley & Sons.

[Amitava Ghosh, January 31, 1997]

I have reviewed this section of the notebook and find it to comply with QAP-001. There is sufficient detail so that another qualified individual could repeat the work described. RG Baca 2/14/97