

**TOTAL-SYSTEM PERFORMANCE ASSESSMENT (TPA)  
VERSION 3.0 CODE:**

**MODULE DESCRIPTIONS AND USER'S GUIDE**

*Prepared for*

**Nuclear Regulatory Commission  
Contract NRC-02-93-005**

*Prepared by*

**Randall D. Manteufel  
Robert G. Baca  
Sitakanta Mohanty  
Mark S. Jarzempa  
Ronald W. Janetzke  
Stuart A. Stothoff  
Charles B. Connor  
Gustavo A. Cragolino  
Asadul H. Chowdhury  
Tim J. McCartin (NRC)  
Tae M. Ahn (NRC)**

**Center for Nuclear Waste Regulatory Analyses  
San Antonio, Texas**

**March 1997**

## ABSTRACT

Total system performance assessment (TSPA) is playing an increasingly important role in regulatory decision-making. Within the U.S. Nuclear Regulatory Commission (NRC) high-level waste program, TSPA studies are being performed to refocus the resource-constrained activities of the NRC Key Technical Issues (KTIs), to evaluate the significance of anticipated changes to the NRC and U.S. Environmental Protection Agency (EPA) regulations, to evaluate the hypotheses of the U.S. Department of Energy (DOE) Waste Containment and Isolation Strategy, and prepare for the NRC review of the DOE viability assessment (VA) for the Yucca Mountain (YM) site.

Conducting a TSPA for the proposed repository site involves the application of a total-system model that simulates the processes affecting repository performance including propagation of the uncertainties associated with model parameters, conceptual models, and future system states. The simulation process, which implements a probabilistic framework, integrates a broad spectrum of site-specific data and information, and produces estimates for a set of regulatory-based performance measures. Building on the previously developed NRC assessment methodology, a new Total-system Performance Assessment (TPA) code, designated TPA Version 3.0 code, was developed for use by the NRC and Center for Nuclear Waste Regulatory Analyses (CNWRA) staff in the planned NRC KTI sensitivity analyses and in assessing the assumptions and models in the forthcoming DOE TSPAs such as the DOE TSPA-VA for the YM site.

The TPA Version 3.0 code is designed to estimate performance measures expected to be specified in NRC and EPA standards. The TPA Version 3.0 code is a combination of an executive driver, a set of consequence modules, and a library of utility modules. The executive driver controls the probabilistic sampling of input parameters, the calculational sequence and data transfers among consequence modules, and the generation of output files. The various output files will be used in parameter importance analyses, post processing of time-dependent risk curves, and synthesis of statistical distributions [e.g., cumulative distribution functions (CDFs) and complementary cumulative distribution functions (CCDFs)] for appropriate performance measures. Consequence modules simulate physical processes and events such as unsaturated zone infiltration, evolution of the near-field thermal-hydrologic environment, corrosion of waste packages, dissolution and release of waste, transport of waste in the groundwater system, extraction of groundwater and consumption of groundwater which if contaminated causes dose to future populations. In addition, disruptive processes such as climate change, faulting, seismicity, and volcanism are modeled. Utility modules ensure the consistency of algorithms and data sets that are used repeatedly by various consequence modules.

This report has been prepared to facilitate the use of the TPA Version 3.0 code by a broad spectrum of users at the NRC and CNWRA. It contains descriptions of:

- Overall TSPA methodology
- Executive or main program controlling overall execution of the code
- Utility modules for general data manipulation and enhancing computational capabilities
- Consequence modules that simulate physical processes and events that affect the release, transport, and evolution of the waste
- Contents and format of the input file (with an example)
- Contents and format of output files
- Regulatory performance measures
- Instructions for program installation and execution

# CONTENTS

Section	Page
FIGURES .....	ix
TABLES .....	xi
ACRONYMS .....	xiii
FOREWORD .....	xv
ACKNOWLEDGMENTS .....	xvii
QUALITY OF DATA AND CODE DEVELOPMENT .....	xvii
1 INTRODUCTION .....	1-1
1.1 REGULATORY BASIS AND PERFORMANCE MEASURES .....	1-1
1.2 BACKGROUND AND PURPOSE OF THE TOTAL-SYSTEM PERFORMANCE ASSESSMENT VERSION 3.0 CODE .....	1-2
1.3 TRANSITION FROM TPA VERSION 2.0 TO 3.0 .....	1-3
1.4 REPORT CONTENT .....	1-5
2 MODEL DESCRIPTION .....	2-1
2.1 OVERVIEW .....	2-1
2.1.1 System Description .....	2-1
2.1.2 System and Disruptive Models .....	2-4
2.1.3 Regulatory Compliance .....	2-4
2.2 THE EXECUTIVE PROGRAM .....	2-5
3 UTILITY MODULES .....	3-1
3.1 READER .....	3-1
3.2 SAMPLER .....	3-3
3.2.1 Constant .....	3-3
3.2.2 Uniform Distribution .....	3-3
3.2.3 Loguniform Distribution .....	3-4
3.2.4 Normal Distribution .....	3-4
3.2.5 Lognormal Distribution .....	3-5
3.2.6 Triangular Distribution .....	3-5
3.2.7 Logtriangular Distribution .....	3-7
3.2.8 Beta Distribution .....	3-7
3.2.9 Logbeta Distribution .....	3-7
3.2.10 Exponential Distribution .....	3-9
3.2.11 Finite Exponential Distribution .....	3-9
3.2.12 User-Defined Distribution .....	3-9
3.2.13 Specified Correlation .....	3-12
3.3 MODULE VARIABLE .....	3-13
3.4 INVENT .....	3-13
3.5 SUBAREA .....	3-16
3.6 ARRAY .....	3-24

## CONTENTS (cont'd)

Section	Page
4	CONSEQUENCE MODULES . . . . . 4-1
4.1	UZFLOW . . . . . 4-1
4.2	NFENV . . . . . 4-4
4.3	EBSFAIL . . . . . 4-14
	4.3.1 Dry Air Oxidation . . . . . 4-15
	4.3.2 Aqueous Corrosion . . . . . 4-16
	4.3.3 Fracture Failure . . . . . 4-18
4.4	EBSREL . . . . . 4-20
4.5	UZFT . . . . . 4-25
4.6	SZFT . . . . . 4-26
4.7	DCAGW . . . . . 4-26
4.8	CLIMATO . . . . . 4-31
4.9	FAULTO . . . . . 4-32
4.10	SEISMO . . . . . 4-33
4.11	VOLCANO . . . . . 4-36
4.12	ASHPLUMO . . . . . 4-40
4.13	ASHRMOVO . . . . . 4-45
4.14	DCAGS . . . . . 4-46
5	INPUT DATA . . . . . 5-1
5.1	INPUT TO TOTAL-SYSTEM PERFORMANCE ASSESSMENT CODE . . . . . 5-1
5.2	FLOW OF DATA BETWEEN MODULES . . . . . 5-1
5.3	DEVELOPMENT OF INPUT FILE <i>tpa.inp</i> . . . . . 5-2
	5.3.1 title . . . . . 5-2
	5.3.2 iflag . . . . . 5-3
	5.3.3 subarea . . . . . 5-4
	5.3.4 aqueousnuclides . . . . . 5-5
	5.3.5 constant . . . . . 5-6
	5.3.6 iconstant . . . . . 5-6
	5.3.7 uniform . . . . . 5-7
	5.3.8 loguniform . . . . . 5-7
	5.3.9 iuniform . . . . . 5-8
	5.3.10 normal . . . . . 5-8
	5.3.11 lognormal . . . . . 5-8
	5.3.12 beta . . . . . 5-9
	5.3.13 logbeta . . . . . 5-9
	5.3.14 triangular . . . . . 5-10
	5.3.15 logtriangular . . . . . 5-10
	5.3.16 exponential . . . . . 5-11
	5.3.17 finiteexponential . . . . . 5-11
	5.3.18 hazardcurve . . . . . 5-12
	5.3.19 userdistribution . . . . . 5-12

## CONTENTS (cont'd)

Section	Page
5.3.20 correlateinputs .....	5-13
5.3.21 endoffile .....	5-13
6 DESCRIPTION OF OUTPUTS .....	6-1
6.1 OVERVIEW .....	6-1
6.2 SAMPLED PARAMETERS .....	6-1
6.3 CONSTANT PARAMETERS .....	6-3
6.4 MODULE VARIABLES (SUBSYSTEM PERFORMANCE MEASURES) .....	6-4
6.5 EPA SUMMED NORMALIZED CUMULATIVE RELEASE .....	6-4
6.6 TIME-DEPENDENT ANNUAL EFFECTIVE DOSE EQUIVALENT .....	6-5
7 PROGRAM INSTALLATION AND EXECUTION .....	7-1
7.1 INSTALLING SOURCE CODE FROM AN 8-MM TAPE .....	7-1
7.2 CUSTOMIZING THE CODE FOR THE USER'S UNIX OPERATING SYSTEM .....	7-1
7.3 PROGRAM EXECUTION .....	7-1
7.4 PORTABILITY LIMITATION OF TPA VERSION 3.0 .....	7-3
7.5 USER SUPPORT .....	7-3
7.6 SOFTWARE QUALITY ASSURANCE .....	7-4
8 REFERENCES .....	8-1
APPENDIX A ORIGINAL SOFTWARE REQUIREMENTS DESCRIPTION FOR TPA VERSION 3.0 CODE	
APPENDIX B EXAMPLE INPUT FILE: <i>tpa.inp</i>	
APPENDIX C DOCUMENTATION OF FUNCTIONS AND SUBROUTINES IN THE UTILITY MODULES	

## FIGURES

Figure	Page
2-1	Overall TSPA flow diagram . . . . . 2-2
2-2	Flow diagram for TPA Version 3.0 code . . . . . 2-6
2-3	Main input(s) and output(s) associated with process modules . . . . . 2-8
2-4	Examples of the three consequence module implementations used in TPA Version 3.0 code: (a) table-lookup, (b) subroutine(s), and (c) stand-alone external program . . . . . 2-9
3-1	Example of uniform distribution . . . . . 3-6
3-2	Example of normal distribution . . . . . 3-6
3-3	Example of triangular distribution . . . . . 3-8
3-4	Example of beta distribution . . . . . 3-8
3-5	Example of exponential distribution . . . . . 3-10
3-6	Example of finite exponential distribution . . . . . 3-10
3-7	Example of user-defined distribution . . . . . 3-11
3-8	Example of specified correlation between two sampled parameters . . . . . 3-14
3-9	Cm-243 decay chain . . . . . 3-17
3-10	Cm-244 decay chain . . . . . 3-17
3-11	Cm-245 decay chain . . . . . 3-18
3-12	Cm-246 decay chain . . . . . 3-19
3-13a	Radionuclide inventories as a function of time (calculated by INVENT module) . . . . . 3-20
3-13b	Radionuclide inventories as a function of time (calculated by INVENT module) . . . . . 3-21
3-13c	Radionuclide inventories as a function of time (calculated by INVENT module) . . . . . 3-22
3-13d	Radionuclide inventories as a function of time (calculated by INVENT module) . . . . . 3-23
3-14	Subarea intersection example . . . . . 3-25
4-1	Example repository discretization into subareas . . . . . 4-2
4-2	Discretization of repository upper block into twelve rectangular regions for the NFENV module . . . . . 4-5
4-3	Mountain-scale heat transfer model with heated rectangular regions . . . . . 4-6
4-4	Thermal network used to predict the maximum spent fuel temperature . . . . . 4-8
4-5	Conceptualization of drift-scale thermal hydrologic model . . . . . 4-12
4-6	Illustration of spent fuel dissolution in a failed waste package . . . . . 4-22
4-7	Schematic drawing for advective and diffusive mass transfer from the waste package to the host rock . . . . . 4-24
4-8	Illustration showing both the unsaturated and saturated zone legs for one subarea . . . . . 4-27
4-9	Example of streamtube with lateral flow (Baca et al., 1996) . . . . . 4-28
4-10	Groundwater dose assessment exposure pathways . . . . . 4-30
4-11	Volcanic scenario implemented in the ASHPLUMO module . . . . . 4-42
6-1	Summary of output files for annual effective dose equivalent . . . . . 6-6

## TABLES

Table		Page
2-1	Current implementation for consequence modules in the TPA Version 3.0 code . . . . .	2-11
3-1	Standard subroutines located in the ARRAY utility module. . . . .	3-26
4-1	Approximate number of water monolayers versus relative humidity (Leygraf, 1995) . . . . .	4-17
6-1	Summary of output files generated by TPA Version 3.0 code . . . . .	6-2

## ACRONYMS

AML	areal mass loading
BWR	boiling water reactor
CCDF	complementary cumulative distribution function
CDF	cumulative distribution function
CLST	Container Lifetime and Source Term
CM	consequence module
CNWRA	Center for Nuclear Waste Regulatory Analyses
CP	compliance point
DBTT	ductile-brittle transition temperature
DCF	dose conversion factor
DEM	digital elevation model
DOE	Department of Energy
DVM	distributed velocity model
EBS	engineered barrier system
EDF	empirical distribution function
ENFE	evolution of the near-field environment
EPA	Environmental Protection Agency
EPRI	Electric Power Research Institute
GWd	gigawatt-day
HLW	high-level waste
IA	Igneous Activity
IPA	Iterative Performance Assessment
KTI	Key Technical Issue
LHS	Latin Hypercube Sampling
MAI	mean annual infiltration
MAP	mean annual precipitation
MAT	mean annual temperature
MPC	multipurpose container
MTU	metric tons of uranium
NAS	National Academy of Sciences
NEA	Nuclear Energy Agency
NRC	Nuclear Regulatory Commission
OECD	Organization for Economic Cooperation and Development
PA	performance assessment
PAAG	Performance Assessment Advisory Group
PDF	probability density function
PWR	pressurized water reactor
QA	quality assurance

## ACRONYMS (cont'd)

RDTME	repository design and thermal-mechanical effects
RH	relative humidity
RT	radionuclide transport
SCP	site characterization plan
SDS	structural deformation and seismicity
SF	spent fuel
SRD	software requirements description
SZ	saturated zone
SwRI	Southwest Research Institute
TEDE	total effective dose equivalent
TEF	thermal effects on flow
THMC	thermal-hydrologic-mechanical-chemical
TOP	Technical Operating Procedure
TPA	Total-system Performance Assessment
TPI	time period of interest
TSPA	Total System Performance Assessment
TSPAI	Total System Performance Assessment and Integration
USFIC	unsaturated and saturated flow under isothermal conditions
UTM	Universal Transverse Mercator
VA	viability assessment
WP	waste package
YM	Yucca Mountain
YMR	Yucca Mountain region

## FOREWORD

In accordance with the provisions of the Nuclear Waste Policy Act of 1982, as amended, the Nuclear Regulatory Commission (NRC) has the responsibility of evaluating a license application for any geological repositories constructed for emplacement of high-level nuclear waste (HLW). This act was amended in 1987 to designate one site in the unsaturated region of tuffaceous rocks of Yucca Mountain in southern Nevada for detailed characterization. The Center for Nuclear Waste Regulatory Analyses (CNWRA) at Southwest Research Institute is a Federally Funded Research and Development Center created to support the NRC in its mission of evaluating and licensing the proposed HLW repository. To meet its licensing function, the NRC will review the application submitted by the U.S. Department of Energy (DOE). One critical section of the license application will deal with the assessment of the future isolation performance of the repository system, which has to achieve certain minimum standards established by federal regulations.

To develop capabilities to review the Total System Performance Assessment (TSPA) in the DOE license application, the NRC and CNWRA are developing and applying performance assessment methods and models using existing site data. Later, at the time of license application review, these methods will be used to conduct an independent TSPA. Because of the large space and time scales involved in estimating repository performance, mathematical models implemented as computer codes are the primary tools for assessing long-term isolation performance. The repository system consists of designed (or engineered) barriers embedded in the natural geological setting. Assessing performance of the total system requires that the behavior of individual barriers be projected for a range of possible future conditions. This assessment process is a complex task that requires a variety of calculations. The Total-system Performance Assessment (TPA) computer code, Version 3.0, is the latest manifestation of software developed for these calculations.

## ACKNOWLEDGMENTS

This report was prepared to document work performed by the Center for Nuclear Waste Regulatory Analyses (CNWRA) for the Nuclear Regulatory Commission (NRC) under Contract No. NRC-02-93-005. The activities reported were performed on behalf of the NRC Office of Nuclear Material Safety and Safeguards, Division of Waste Management. The report is an independent product of the CNWRA and does not necessarily reflect the views or regulatory position of the NRC.

The authors wish to thank M.S. Jarzempa, CNWRA, and R.W. Rice, Portage Environmental, Inc., for their thorough technical reviews and numerous helpful technical comments. The authors also thank B. Sagar and W. Patrick for their programmatic reviews and the valuable suggestions that improved the quality of the final report. C. Garcia has earned a special recognition for her skillful and timely secretarial efforts. In preparing this document and in developing the TPA Version 3.0 code, a number of individuals made strong technical contributions to the effort. Thanks are expressed to J.R. Weldy (CNWRA), R.W. Rice, K.J. Poor, and D.J. Thorne of Portage Environmental, Inc. for their assistance in conducting and documenting the module testing. A number of CNWRA staff made contributions through their participation in technical discussions that lead to module formulations and input data. These contributors were J.A. Stamatakos, B.E. Hill, P.C. Lichtner, W.M. Murphy, G.W. Wittmeyer, S.M. Hsiung, N. Sridhar, A. Ghosh, P.A. LaPlante, R.T. Pabalan, R.T. Green, D.R. Turner, E.C. Percy, M.P. Ahola, M.P. Miklas, and A.R. DeWispelare. The NRC technical staff also made important contributions to the TPA Version 3.0 code. In particular, R.G. Wescott, R.B. Codell, N.A. Eisenberg V.A. Colten-Bradley, D.W. Vinson, K.A. Gruss, J.R. Firth, P.S. Justus, and C.A. McKenney made contributions to the development of the conceptual and mathematical models implemented in the TPA Version 3.0 code, and some also for testing of the initial version of the code. Their contributions are acknowledged and much appreciated.

## QUALITY OF DATA AND CODE DEVELOPMENT

**DATA:** CNWRA-generated laboratory data contained in this report meet quality assurance (QA) requirements described in the CNWRA QA Manual. Data from other sources, however, are freely used. The respective sources of non-CNWRA data should be consulted for determining their levels of quality assurance.

**CODE:** The TPA Version 3.0 code has been developed following the procedures described in CNWRA Technical Operating Procedure, TOP-018 which implements the QA guidance contained in the CNWRA QA Manual.

# 1 INTRODUCTION

The Total-system Performance Assessment (TPA) code, designated as Version 2.0 (Sagar and Janetzke, 1993), was developed for, and extensively used in, the Nuclear Regulatory Commission/Center for Nuclear Waste Regulatory Analyses (NRC/CNWRA) Iterative Performance Assessment (IPA), Phase 2, of the proposed high-level waste (HLW) repository at Yucca Mountain (YM), Nevada. This document describes the new TPA code, Version 3.0, which is intended for use in future evaluations of the proposed HLW repository site. The new TPA Version 3.0 code simulates the processes affecting repository performance taking into account the uncertainties in model parameters, conceptual models, and future system states. The simulation process, which integrates a broad spectrum of site specific data and information (e.g., site characterization data, engineered barrier designs, and biosphere data), produces probabilistic estimates for regulatory-based performance measures.

The TPA Version 3.0 code was developed within the NRC/CNWRA Total System Performance Assessment and Integration (TSPAI) Key Technical Issue (KTI). However, the knowledge base and models produced by other KTIs have been integrated into the code. In addition, the new TPA code accommodates updated aspects of the repository program such as (i) the latest U.S. Department of Energy (DOE) repository layout, waste package (WP) and emplacement design; (ii) current and anticipated standards (release-based and risk-based); and (iii) arbitrary compliance period (thousands to hundreds of thousands of years). The TPA Version 3.0 code is expected to be used across the NRC HLW program to:

- Determine the relative importance of individual KTIs to total-system performance
- Examine implementability of anticipated changes to the NRC regulations and U.S. Environmental Protection Agency (EPA) standards
- Evaluate the hypotheses of the DOE Waste Containment and Isolation Strategy (U.S. Department of Energy, 1996)

In addition, the TPA Version 3.0 code will be used in probing the vulnerabilities of future DOE TSPAs such as the forthcoming viability assessment (TRW Environmental Safety Systems, Inc., 1996) for the YM site.

This document was prepared to facilitate the use of the TPA Version 3.0 code by a broad spectrum of NRC and CNWRA staff. The report contains descriptions of the various modules, keywords used in preparing the input file, instructions on code installation and execution, and an example input file.

## 1.1 REGULATORY BASIS AND PERFORMANCE MEASURES

A specific regulatory purpose for conducting a Total System Performance Assessment (TSPA) is to determine whether the proposed repository system complies with the applicable environmental standards. This regulatory determination is generally the result of a complex and highly intricate engineering analysis that ultimately culminates in comparing the results of performance measure calculations with the regulatory standards. The TPA Version 3.0 code is designed to estimate performance measures that are currently or expected to be specified in NRC regulations and EPA standards, including (i) container lifetime, (ii) release rate from the engineered barrier, (iii) groundwater travel time, (iv) cumulative release, and (v) peak dose to an individual. In addition, the TPA Version 3.0 code will permit the performance assessment (PA) analyst to ascertain the dependence of the performance measures on individual repository components and site characteristics, and model abstractions.

The primary regulations applicable to the proposed geological repository for HLW were promulgated by the NRC in 10 CFR Part 60—Disposal of High-Level Radioactive Wastes in Geologic Repositories. Two sections of 10 CFR Part 60 pertain specifically to postclosure performance. These sections are 10 CFR 60.112—Overall System Performance Objective for the Geologic Repository after Permanent Closure; and 10 CFR 60.113—Performance of Particular Barriers after Permanent Closure. Section 60.112 makes reference to satisfying the generally applicable environmental standards for radioactivity established by the EPA (e.g., 40 CFR Part 191). Performance measures used in 10 CFR 60.113 to define performance of individual barriers (in contrast to the total system) are: (i) life of the WP must exceed specified limits [10 CFR 60.113(a)(ii)(A)—Substantially Complete Containment Requirement], (ii) release from engineered barriers must be less than specified limits [10 CFR 60.113(a)(1)(ii)(B)—Groundwater Release Requirement], and (iii) groundwater travel time must be greater than specified limits [10 CFR 60.113(a)(2)—Groundwater Travel Time Requirement].

In the Energy Policy Act of 1992, the U.S. Congress directed the National Academy of Sciences (NAS) to make recommendations about an environmental standard for the proposed geologic repository at YM. In its recommendations (National Academy of Sciences, 1995), the NAS proposed that the new environmental standard, to be issued by the EPA, be based on peak risk to the average member of a small, critical group. The NRC is currently enhancing its capability to evaluate the DOE TSPAs (including the expected in 1998 and license application in 2002) by developing the TPA Version 3.0 code.

## **1.2 BACKGROUND AND PURPOSE OF THE TOTAL-SYSTEM PERFORMANCE ASSESSMENT VERSION 3.0 CODE**

One of the purposes of conducting a TSPA is to secure a detailed understanding of: (i) the key factors controlling the degradation of the engineered barrier system (EBS), (ii) release of the waste from the repository, (iii) subsequent transport of the waste through various environmental pathways, and (iv) possible human exposure at the location(s) of the critical group. To achieve this understanding, the total repository system is modeled in a probabilistic manner (Thompson and Sagar, 1993) that considers significant physical and chemical processes, phenomenological interactions and couplings, and potentially disruptive events and processes. This probabilistic approach, although computationally intensive, yields a range of potential future evolutions of the repository system. In addition, this approach is widely favored because it avoids many of the technical shortcomings associated with completely deterministic scenario-based assessments (Thompson, 1988).

Two NRC TSPAs for YM have been completed to date using the probabilistic approach. The first TSPA, referred to as IPA Phase 1 (Codell et al., 1992), was conducted to assemble and demonstrate the NRC assessment methodology. The second TSPA, designated as IPA Phase 2 (Wescott, et al., 1995), broadened the assessment framework, produced the initial version of the TPA code, and yielded considerable insight into processes influencing repository performance. In IPA Phase 3 (Manteufel et al., 1995), emphasis was placed on updating and advancing the NRC TSPA capability for future use in the review of the DOE TSPA-VA (TRW Environmental Safety Systems, Inc., 1996) as well as for focusing the NRC HLW program activities on technical issues of critical importance to repository performance.

Consistent with the recommendations of IPA Phase 2 (Wescott, et al., 1995), the new TPA Version 3.0 code was developed to include: (i) updated abstractions for process and consequence modules, (ii) improved capabilities for parameter importance analysis and ranking, (iii) site specific data and bases for consequence modules, (iv) streamlined methodology for data transfers between the executive and

consequence modules, (v) more flexible design to accommodate future code modifications, and (vi) improved computational algorithms to the extent practical. Because regulations governing disposal are still under development, site data continue to be acquired, and the DOE repository design is evolving, it was deemed vital to develop the TPA code with the flexibility to:

- Increase or decrease the time period of interest (TPI)
- Use finer spatial discretizations of the repository layout
- Use finer time discretizations with uniform or nonuniform steps
- Specify different areal mass loading (AML)
- Assess the effect of various disruptive scenarios
- Calculate time-dependent dose rate at a compliance point (CP)
- Calculate peak dose rate at a CP in the specified TPI
- Assess the impact of radionuclide dilution in the saturated zone (SZ)
- Accommodate the current in-drift emplacement design
- Assign statistical distributions to uncertain model parameters
- Assess different model abstractions in the consequence modules
- Evaluate alternative repository and waste package design features

The TPA Version 3.0 code is a combination of an executive driver, a set of consequence modules, and a library of utility modules. The executive driver controls the probabilistic sampling of input parameters, the calculational sequence and data transfers between modules, and the generation of output files. Various output files are generated for later use in parameter importance analyses, post-processing of time-dependent risk curves, and synthesis of statistical distributions [e.g., cumulative distribution function (CDF) and complementary cumulative distribution functions (CCDFs)] for appropriate performance measures. Utility modules ensure that consistent data sets are used by all consequence modules and facilitate the discretization of the repository system and surrounding geologic media.

### **1.3 TRANSITION FROM TPA VERSION 2.0 TO 3.0**

The TPA computer code designated as Version 2.0 (Sagar and Janetzke, 1993) was developed in 1993 for the NRC/CNWRA IPA Phase 2 exercise (Wescott et al., 1995) of the proposed YM repository site. The TPA Version 2.0 code was designed to calculate the performance measures specified in NRC 10 CFR Part 60 (i.e., container lifetime, release rate from the EBS, and groundwater travel time) and EPA 40 CFR Part 191 (i.e., cumulative release for 10,000 yr). Additionally, the code calculated the integrated population dose over 10,000 yr. Since its initial development, extensive experience has been gained while using and attempting to adapt the code to changing regulatory and programmatic environments. Specific recommendations in IPA Phase 2 were made for improving the TPA code (Wescott et al., 1995), including:

- Software quality assurance (QA) requirements need to be more prominent in module development
- Future IPA development will require more model abstractions and efficient computing techniques
- The TPA code must be easily upgraded

In developing the TPA Version 3.0 code, considerable effort was devoted to implementing these recommendations, particularly to those related to software QA, addition of new modules, uniform

interfaces between modules, and uniform coding practices among modules. With regard to the third recommendation, Wescott et al. (1995) recommended that the TPA system code be considered a dynamic entity, to be upgraded in future IPA iterations.

Major revisions to the TPA executive and consequence modules have been made, hence the designation of a new version number. These revisions were required not only from a software viewpoint, but also because the regulatory environment has changed in recent years. Specifically, code changes have been incorporated to address: (i) recommendations of the NAS (National Academy of Sciences, 1995) to the EPA to develop new performance criteria for YM, (ii) substantial changes in the DOE repository and WP design concepts (U.S. Department of Energy, 1996), (iii) expanded knowledge base, improved models, and additional data compiled by both DOE- and NRC-funded investigations, and (iv) increased needs by the NRC staff to perform KTI specific sensitivity analyses.

The NAS recommendations have necessitated TPA code changes involving: (i) addition of a new performance measure (individual annual dose), (ii) addition of variable compliance period (i.e., time to peak dose not necessarily 10,000 yr), (iii) separate treatment of human intrusion by drilling (i.e., not to be considered as part of the main compliance determination; this feature may be added to be able to explore the impact of proposed changes to EPA HLW standards and NRC HLW regulations), and (iv) incorporation of more appropriate representations of the environmental pathways and biosphere.

Recent changes in the DOE repository and WP design concepts have also necessitated updating the TPA Version 3.0 code to account for: (i) larger WPs, (ii) in-drift emplacement, (iii) higher areal mass loading, (iv) changing location and footprint of the repository, (v) changing drift support and backfill emplacement concepts, and (v) prolonged operational time period prior to permanent closure of the facility. The NRC staff have a regulatory interest in assessing the impact of these design changes from a total-system perspective; consequently, these features need to be a part of the TPA code.

Concurrent with changes in design, ongoing site characterization and research programs have provided new data and models, as well as an improved understanding of the YM site. Specifically, new models have been realized in the areas of infiltration, deep percolation, and unsaturated zone flow (e.g., Stothoff et al., 1996). More current field data and conceptual models have been employed to dynamically simulate these processes. Technical contributions from the various KTIs have been instrumental in enhancing models in the new TPA Version 3.0 code. Some of the noteworthy contributions include:

- New WP failure and source term models from the Container Lifetime and Source Term (CLST) KTI
- Improved technical basis for the probability and scenario characteristics for volcanic activity from the Igneous Activity (IA) KTI
- Updated estimates of elemental solubilities and sorption coefficients (Kd) from the Radionuclide Transport (RT) KTI
- Improved hydrostratigraphic model and technical specification for a new faulting disruptive consequence modules from the Structural Deformation and Seismicity (SDS) KTI
- Improved capability to predict time-dependent temperature and relative humidity for use in the source term module from the Thermal Effects on Flow (TEF) KTI
- Estimates of groundwater chemical composition from the Evolution of the Near-Field Environment (ENFE) KTI

- Improved modeling capability for a new seismic consequence module from the Repository Design and Thermal-Mechanical Effects (RDTME) KTI
- Improved information on climate and the distribution of infiltration over the repository area from the Unsaturated and Saturated Flow under Isothermal Conditions (USFIC) KTI

The intended use and anticipated users of the new TPA Version 3.0 code also have changed. Unlike the TPA Version 2.0 code which was difficult for a new user to grasp and master, the TPA Version 3.0 code was designed to provide a more intuitive input structure and input file error traps, as well as a capability to provide helpful diagnostics to the user. The overall code has been designed to be much more transparent (i.e., easy to understand and to follow the coding logic), so that it can be used in planned KTI sensitivity studies by non-TPA developers. The code architecture permits the user to easily understand the data and calculational flow between modules and allows the user direct access to inputs and outputs for the conduct of sensitivity analyses.

In summary, the basic rationale for creating a new version of the TPA code was: (i) the need for a code suitable for use by a wide variety of users, (ii) the requirement to accommodate expected changes to the applicable regulations, (iii) the need to accommodate substantial changes in the repository and WP design concepts, and (iv) the need for flexibility to incorporate new and improved data and models. Because the TPA code is to serve a broad spectrum of users, the new version of the code has been designed to be highly transparent with regard to its functionality, significantly easier to modify and use, and computationally more efficient. In developing the TPA Version 3.0 code, the consequence modules relevant to the new regulatory requirements have been placed under a new and more compact executive program. The consequence and utility modules that have been developed or modified have undergone basic software QA testing.

## **1.4 REPORT CONTENT**

A brief description of the executive module is contained in chapter 2. The various utility modules are described in chapter 3, while the consequence modules are presented in chapter 4. Chapter 5 describes the input files and parameters for the TPA Version 3.0 code, and chapter 6 describes the outputs. Program installation and execution are described in chapter 7, while chapter 8 lists the references. The original software requirements description (SRD) for the TPA Version 3.0 code is included in appendix A. An example input file is contained in appendix B and documentation of the utility module functions and subroutines is provided in appendix C.

## 2 MODEL DESCRIPTION

### 2.1 OVERVIEW

The TPA Version 3.0 code implements the NRC TSPA methodology presented by Wescott et al. (1995) and outlined in figure 2-1. There are four major components of the TSPA methodology:

- System characterization
- System models for anticipated processes and events
- Disruptive models for processes initiated by external events
- Subsystem performance measures
- Total system performance and regulatory compliance assessment

#### 2.1.1 System Description

A TSPA analysis starts with system characterization and ends in the assessment of subsystem performance and regulatory compliance determination. System characterization is a continual process that involves gathering data for site characteristics, waste form properties, waste package design, repository design, and biosphere characteristics. Examples of system characterization include:

- Representation of YM region hydrostratigraphy
- Determination of properties for distinct hydrothermal and hydrostratigraphic units
- Characterization of saturated zone flow rates, direction, and dispersion
- Identification and characterization of fault zones in the YM region
- Characterization of past volcanic events in the YM region
- Characterization of past seismic events in the YM region

The HLW constitutes the hazard associated with the disposal facility and needs to be characterized. Examples of waste property characterization include:

- Total amount of waste for disposal [e.g., 70,000 metric tons of uranium (MTU)]
- Variety of waste sources [e.g., from commercial Boiling Water Reactors (BWR), from commercial Pressurized Water Reactors (PWR), and from defense weapons production reactors and the subsequent extraction of fissile material]
- Waste forms (e.g., intact spent fuel assemblies and vitrified waste)
- Age of waste from the reactor to repository emplacement
- Burnup of waste (in the case of spent nuclear fuel)
- Inventory of key radionuclides in the waste that pose long-term health hazards

The radioactive waste will be emplaced using disposal packages designed to contain the waste for extended periods of time following emplacement. Recently, the WP design has evolved and changed substantially. For example, in the Site Characterization Plan (SCP) (U.S. Department of Energy, 1988), the concept was to use thin-walled stainless steel containers that each hold about 2.5 MTU of waste. Current designs (U.S. Department of Energy, 1996) include larger packages that contain about 8.8 MTU of waste and have two thick walls of 10-cm outer corrosion allowance material (carbon steel) and 2-cm inner corrosion resistance material (stainless steel). Because long-term containment of the waste in engineered packages, potentially exceeding 10,000 yr (e.g., TRW Environmental Safety Systems,

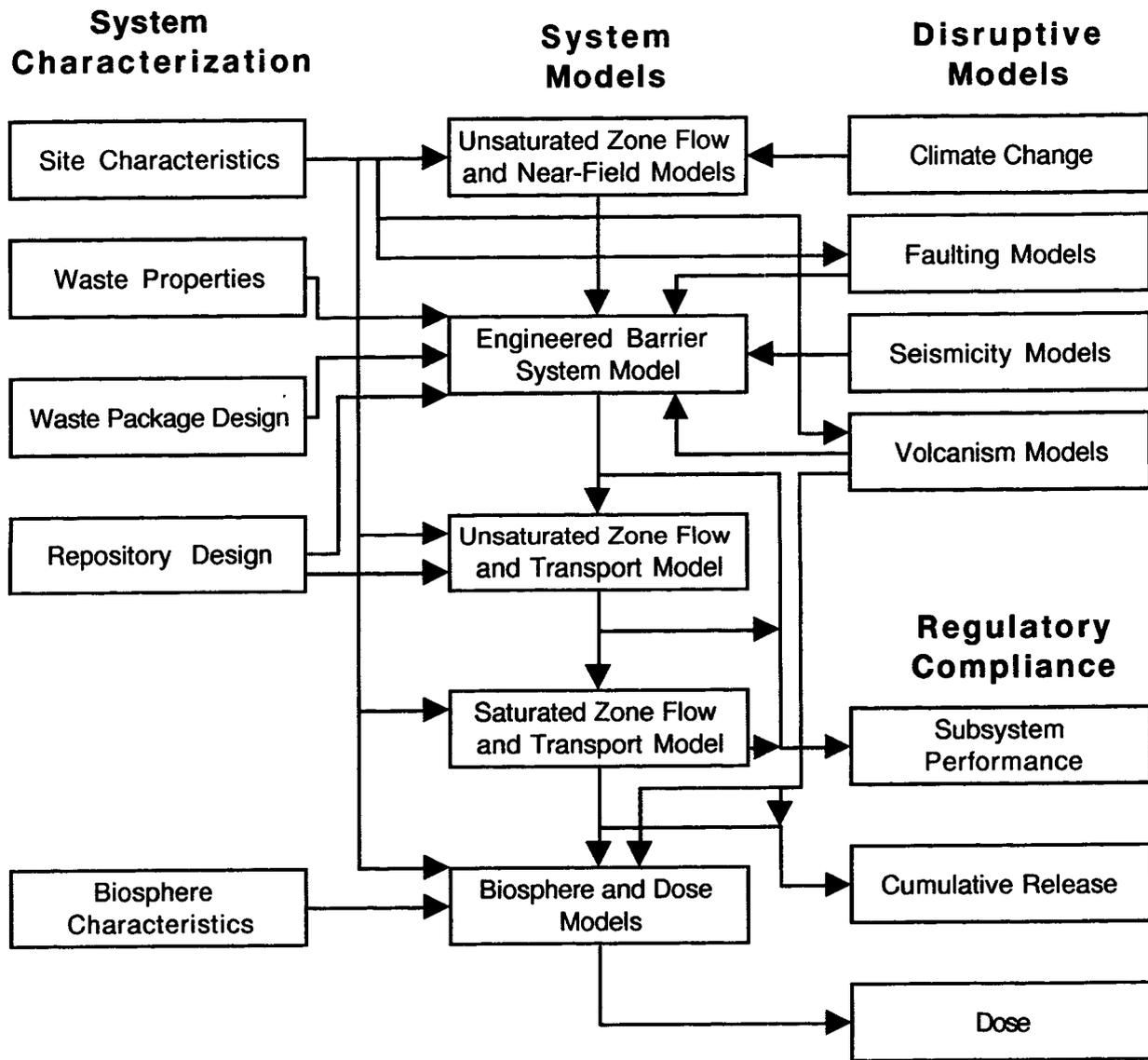


Figure 2-1. Overall TSPA flow diagram

Inc., 1995) will be postulated, characterization of the waste package design is important to a TSPA. Examples of the waste package design characterization include:

- Dimensions of the package
- Properties (e.g., thermal, mechanical, and corrosion resistance) and thickness of the walls of the package
- Payload of the package (e.g., 8.8 MTU)

Similarly, the DOE repository design concept has changed significantly over the past few years. Many of these changes were motivated to control the anticipated thermal-hydrologic-mechanical-chemical (THMC) environment of waste packages. The near-field environment can have a strong influence on the waste package lifetime. In the SCP (U.S. Department of Energy, 1988), the packages were to be placed in vertical boreholes beneath the floor of emplacement drifts. The new repository concept includes in-drift emplacement of the waste packages (U.S. Department of Energy, 1996). Examples of the repository design characterization include:

- Orientation and location of WPs at emplacement
- Drift and WP spacing [this affects the areal mass loading (AML) of the waste]
- Use of concrete inverts, and steel or concrete drift supports
- Potential use of ventilation during the operational time period of the facility
- Potential emplacement of a backfill material after the operational time period
- Location and permanent sealing of shafts, ramps, and boreholes following the operational time period

Another component of the system to be characterized is the biosphere. Previously, EPA regulations were based on a cumulative release standard derived from generic considerations of biosphere models. Recently, the NAS (National Academy of Sciences, 1995) has recommended that the regulatory standards applicable to YM be changed from a release-based standard to a risk-based standard. Although new regulations have not been promulgated, these recommendations are expected to be adopted. To compute risk, it is necessary to model the biosphere and estimate health effects, which are frequently expressed in terms of annual dose. Characterization of a biosphere model at YM has many factors, including:

- Population locations (i.e., designated critical groups)
- Land-type and use (e.g., farming, ranching)
- Availability of groundwater for domestic and agricultural use
- Agricultural plant and animal types
- Human exposure pathways

In conducting a TSPA, one must integrate a wide variety of site, waste type, package design, repository design, and biosphere characteristics. The goal of a TSPA, however, is to focus on a subset of the most important characteristics. Much national (e.g., Davis et al., 1990) and international [e.g., the Performance Assessment Advisory Group (PAAG) of the Nuclear Energy Agency (NEA) of the Organization for Economic Cooperation and Development (OECD)] effort has been devoted to developing systematic procedures in which a wide spectrum of experts representing a diversity of disciplines can develop and agree on a balanced subset of important characteristics that should be included in a TSPA. For waste properties and WP and repository designs, this subset of characteristics is more easily established. For site characteristics associated with low-probability disruptive events, methods for scenario

identification have been developed (e.g., Cranwell et al., 1990). In the two previous NRC TSPA exercises (Codell et al., 1992; Wescott et al., 1995), a scenario selection methodology was employed and found to be adequate for these exercises; however, scenario assumptions should continually be reviewed and refined to ensure that all important features, events, and processes have been considered (Bonano and Baca, 1994).

### **2.1.2 System and Disruptive Models**

Closely related to system characterization is model development. For analysis purposes, models have been classified into those for either anticipated processes (also called base-case processes) or disruptive processes. The primary distinction is the source which causes a response in the system. If the source is the emplaced HLW, then the processes are anticipated. If the source is external then the process is classified as disruptive. An example of an anticipated process is the thermal-hydrologic response caused by the emplacement of heat-dissipating waste. An example of a disruptive process is seismic activity caused by regional tectonic and geologic processes not related to the repository.

In figure 2-1, the system model has several major process models, including:

- Unsaturated zone flow and near-field environment of the EBS
- Corrosion and other anticipated failure mechanisms of the EBS containment
- Groundwater flow and radionuclide transport in the unsaturated zone below the proposed repository and into the saturated zone
- Groundwater flow and radionuclide transport in the saturated zone below the proposed repository to a compliance point (CP) or boundary
- Radionuclide transport in the biosphere through the groundwater pathway that ultimately results in a dose to humans

Specific implementations of these process models are described in chapter 4 of this report, hence, they are only mentioned here.

The disruptive models include:

- Climate change leading to changes in groundwater percolation rates at YM
- Faulting leading to disruption of emplacement drifts and alterations along fault zones
- Seismicity leading to rockfall and disruption of emplacement drifts
- Volcanism leading to magma contacting WPs and depositing contents on the ground surface

Climate change, faulting, seismicity, and volcanism can lead to earlier failures of the EBS containment and alter the character of the natural system. In the case of volcanism, radionuclides may be released directly into the biosphere through extrusive events that disperse contaminated volcanic ash over the ground surface.

### **2.1.3 Regulatory Compliance**

The ultimate goal of a TSPA is to gain insights into the important characteristics and processes that influence system compliance with regulatory standards. Currently, this understanding includes calculating the cumulative release of radionuclides (at a compliance boundary) and assessing dose-based risk to critical population groups. In addition, numerous subsystem performance measures have been used

and are expected to continue to be used by the NRC to develop a more in-depth understanding of the repository system performance. These subsystem performance measures include groundwater travel time, waste package lifetime, and nuclide release rates from the EBS.

## 2.2 THE EXECUTIVE PROGRAM

The Executive (EXEC) or main program controls the overall code execution, including reading the input data file, controlling the flow of data between various modules, sampling of input parameters, identifying and initiating consequence modules, and generating output data. The EXEC controls the execution of numerous subroutines and simulation codes that are categorized as either utility modules or consequence modules. Utility modules provide general data storage and retrieval capabilities, as well as generic computational capabilities. This modularization is consistent with the software design principles of decomposition and abstraction (Liskov and Guttag, 1986). Many benefits result from having utility modules, including improved code utilization and reliability. The primary motivation for utility modules is to minimize the potential for errors in common calculations (e.g., reading input data, adding arrays, solving equations, calculating radionuclide inventories, and sampling parameters). Consequence modules simulate the physical processes and events that lead to release, transport, and evolution of the waste. Briefly summarized, consequence modules simulate physical processes specific to the YM repository system using the utility modules as needed. The utility modules are described in chapter 3, and the consequence modules are described in chapter 4 of this report. The software requirements description that originally outlined the TPA Version 3.0 code design is provided in appendix A.

Through the use of utility modules, the EXEC controls the spatial discretization of the proposed repository (i.e., number of subareas), the distance from the proposed repository to the CP or boundary (e.g., 5, 20, 25, or 30 km), the temporal discretization scheme (e.g., output every 200 yr), and the time period of interest (e.g., 10,000 yr). These data, as well as other data, are specified by the user in the *tpa.inp* file. An example of a *tpa.inp* file is provided in appendix B.

As illustrated in figure 2-2, the EXEC controls the flow of data to and from the consequence modules. The EXEC begins the simulation by using the READER utility module to read the *tpa.inp* file. The information in the *tpa.inp* file can be extensive because it is designed to be the sole source of user specified data for the entire TPA Version 3.0 code. User input data are those data that may be changed by the user for a given application. In contrast, static data refers to fixed parameter values set within the TPA Version 3.0 code. For example, the digitized elevation of the ground surface in the vicinity of YM is stored in a static data file. The static data file is archived so that it is part of the TPA library, just like utility and consequence modules. Hence, static data are a part of the TPA Version 3.0 code.

The READER needs to be called only once during a run to read the dynamic data from the *tpa.inp* file. Having read and stored the input parameters, the EXEC continues by executing consequence modules. The process modules include: UZFLOW, NFENV, EBSFAIL, EBSREL, UZFT, SZFT, and DCAGW. During the run, all the process consequence modules are executed. Based on input specified in the *tpa.inp* file, the EXEC also controls which disruptive consequence modules will be executed. The disruptive modules include: CLIMATO, FAULTO, SEISMO, VOLCANO, ASHPLUMO, ASHRMOVO, and DCAGS.

The EXEC interfaces with each of the consequence modules using a subroutine call statement which has explicit input and output expectations. This convention places very few restrictions on the

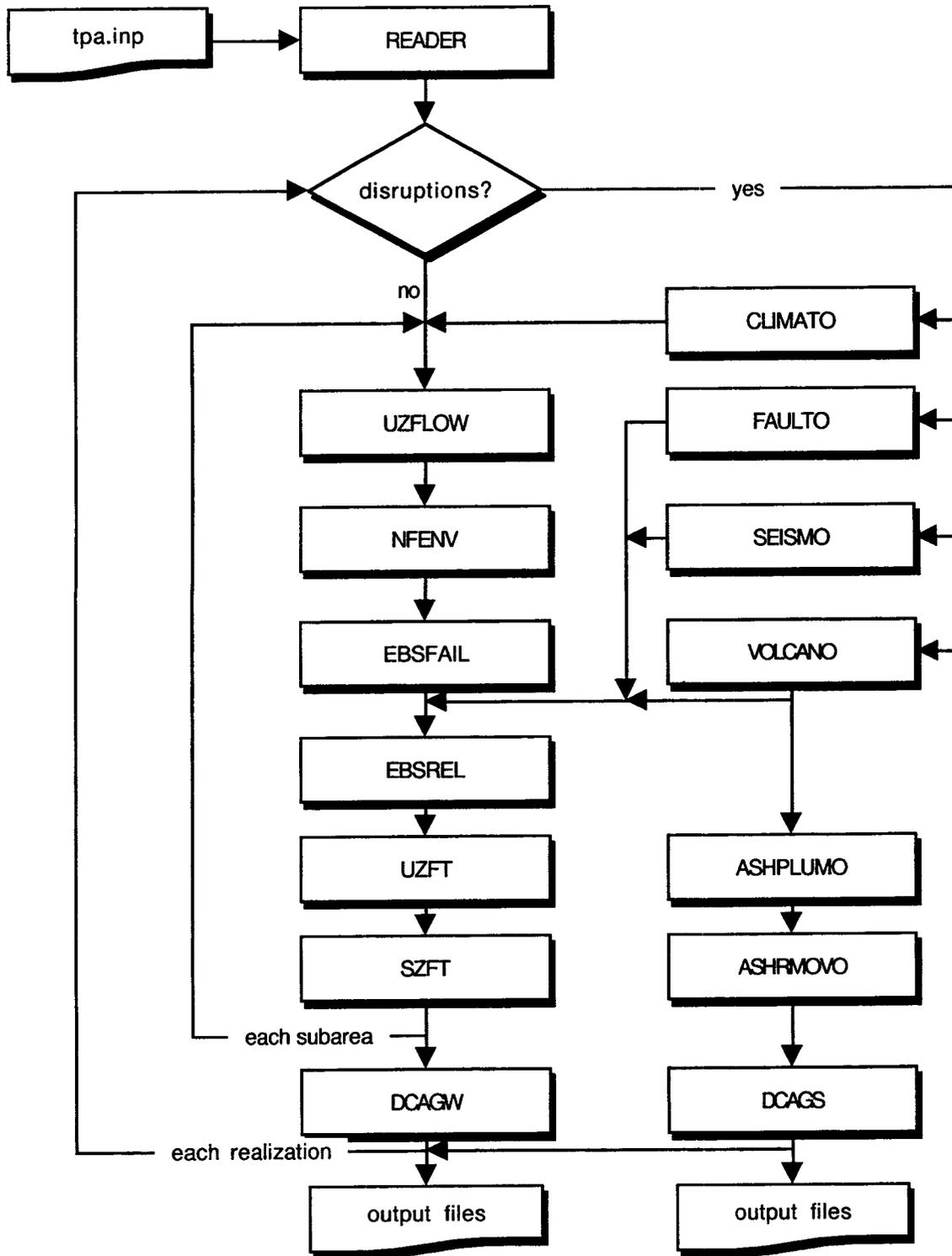


Figure 2-2. Flow diagram for TPA Version 3.0 code

internal structure and functionality of individual consequence modules, yet greatly enhances the modularity of the code. The parameters and the arrays being passed in the call statement have been specified for each of the consequence modules. This is consistent with the software design principle of procedural abstraction (Liskov and Guttag, 1986). Only the input and output expectations for each module are explicitly specified, and module developers can implement the most appropriate models for their intended purpose as long as the specifications are upheld.

The primary inputs and outputs associated with the process modules are illustrated in figure 2-3. The input and output variables in the illustration have names that are as descriptive as possible. The inputs and outputs are the primary time-dependent data expected to change from realization to realization. Typically the output of one module is input to the next module in the calculational process, yet certain output may be required by more than one module. As shown figure 2-3, the calculational process starts with the UZFLOW module, which is expected to return the time-dependent flow rate into a particular subarea per WP in the subarea. The units of the output are cubic meters of water per year. These results are the input to the NFENV module, which outputs time-dependent solutions for: (i) flow rate into the subarea that does not contact the WP, (ii) flow rate into the subarea that contacts the WP, (iii) temperature of the rock wall near the WP, (iv) temperature of the WP surface, (v) maximum temperature of the spent fuel in the WP, (vi) relative humidity at the WP surface, (vii) pH of the ground water contacting the WP (single value assumed), and (viii) chloride concentration in the ground water contacting the WP. This information is required by the EBSFAIL code, which then predicts the time-dependent percent corrosion failure of the WP. Transferring data between the EXEC and consequence module in this manner is very efficient because in the FORTRAN language when an array is passed, only the identification (i.e., pointer) is actually passed. The last consequence module is DCAGW, which calculates the annual total effective dose equivalent per nuclide in the groundwater pathway in units of rem.

Many viable implementations can be used within a consequence module, but only three distinct approaches have been used in the TPA Version 3.0 code. The first implementation is a simple one in which the consequence module reads from one or more static data files. Figure 2-4(a) illustrates this implementation. It is commonly referred to as "table-lookup" and is often used when a stand-alone program requires excessive computing times to yield results. In this situation the stand-alone program is run off-line and the results are stored in a static lookup file. An example of the use of table-lookup is the abstraction of numerical results from UDEC (Itasca Consulting Group, Inc., 1996) simulations for the SEISMO module. The program UDEC requires many hours to complete one dynamic simulation of an emplacement drift subjected to a seismic load. Consequently, the results of many simulations have been stored in a static data file, so that these results can be retrieved and used within the TPA code.

The second method of implementation illustrated in figure 2-4(b) is the use of one or more subroutines. Here, calculations are performed without reliance on static data files or stand-alone programs. The module consists of a suite of numerical models in subroutines that the analyst has selected to use. Examples of this approach include DCAGW and ASHRMOVO. These modules were created specifically for the TPA Version 3.0 code, and have incorporated specific conceptual and mathematical models.

The third approach of implementation illustrated in figure 2-4(c) is execution of a stand-alone computer program. The consequence module usually writes (or modifies) one or more input files required by the stand-alone program. Then, a nonstandard FORTRAN call is made within the module to spawn an external process that runs the stand-alone program. Although this feature is nonstandard FORTRAN, it is provided in many compilers, including those on Sun computers (SunSoft, Inc., 1996). This feature appears to be unavailable on compilers for personal computers using DOS or Windows operating systems. If this feature is not available, then the TPA Version 3.0 code will not be able to be run on those

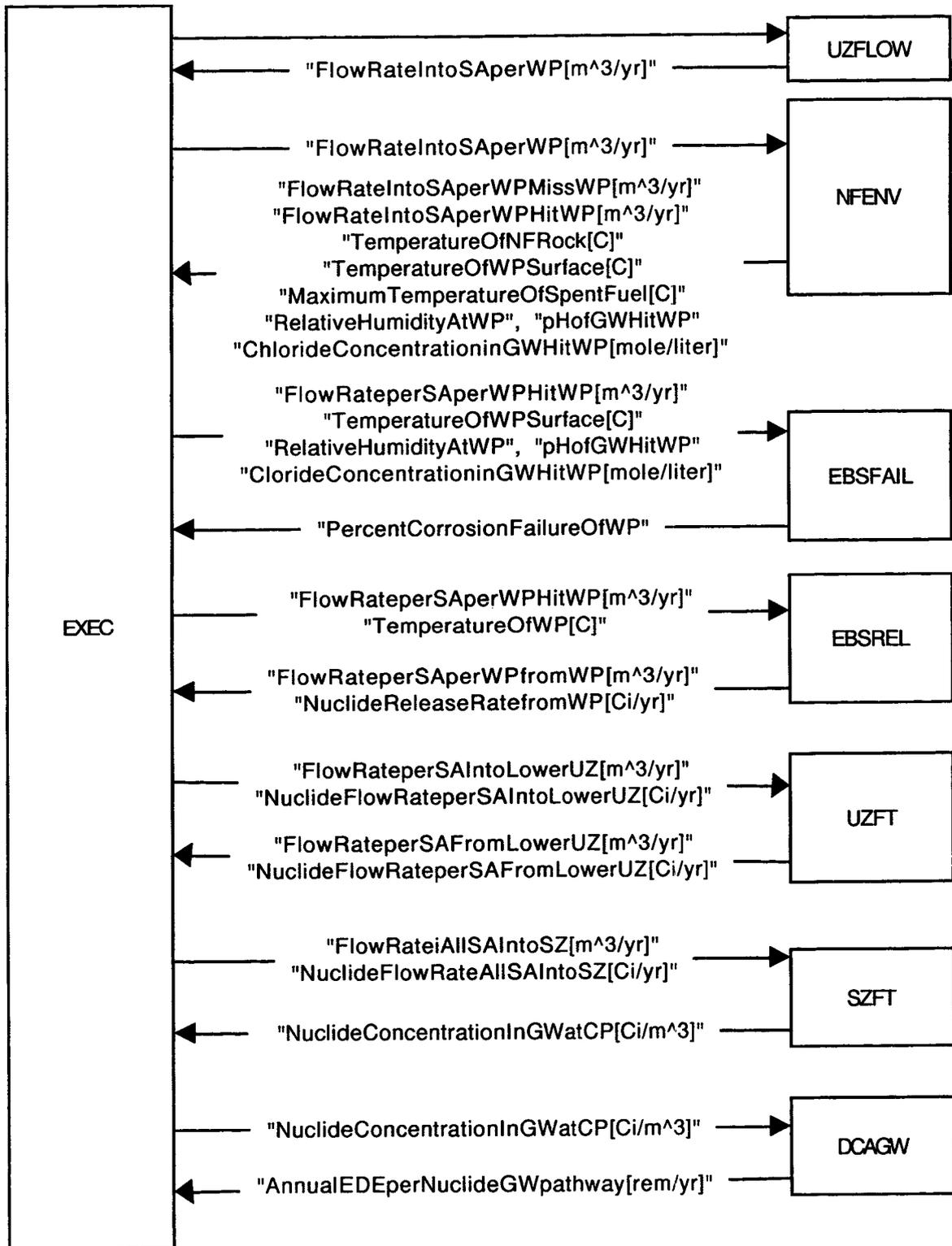
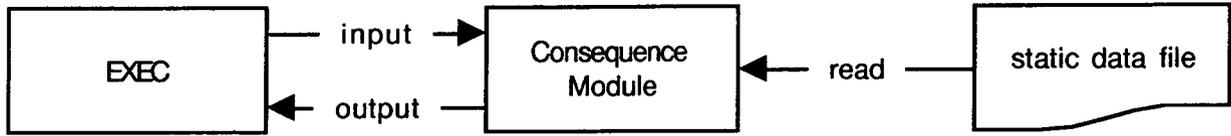
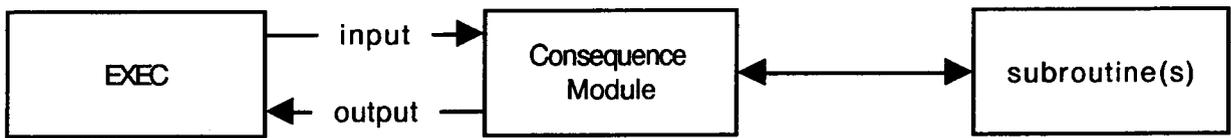


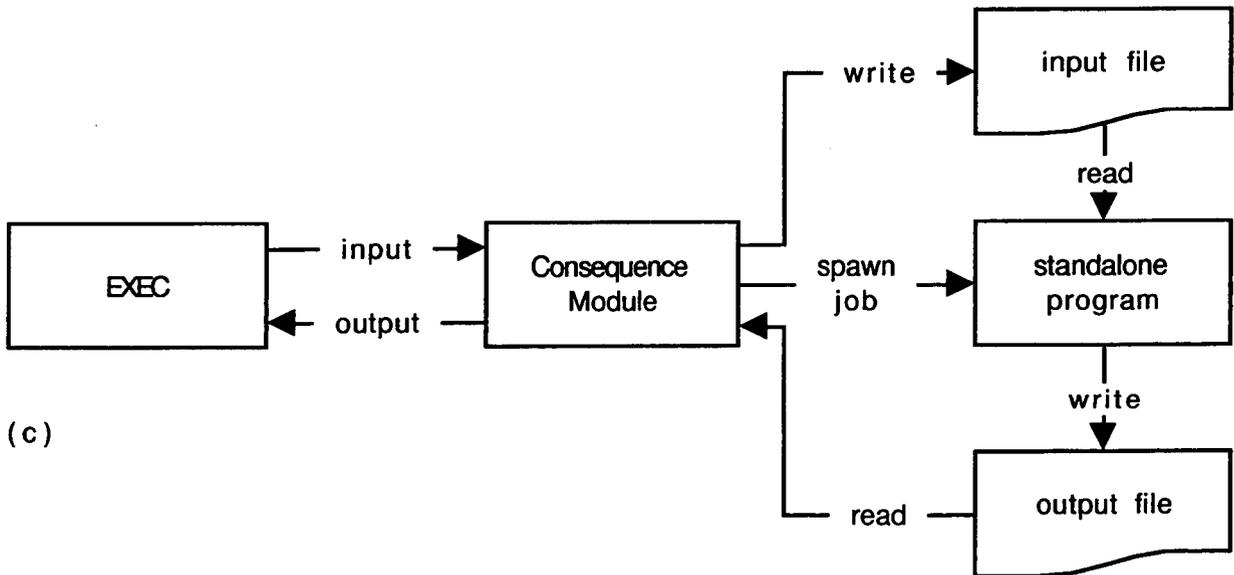
Figure 2-3. Main input(s) and output(s) associated with process modules



(a)



(b)



(c)

**Figure 2-4. Examples of the three consequence module implementations used in TPA Version 3.0 code: (a) table-lookup, (b) subroutine(s), and (c) external stand-alone program**

platforms, and portability will not be possible. This restriction is not, however, a consequence of the overall design of the TPA Version 3.0 code or the EXEC, but it is a consequence of choosing this implementation for the consequence modules. Currently, it appears the TPA Version 3.0 code will be restricted to applications on computers with UNIX operating systems. This same restriction also existed for the TPA Version 2.0 code (Sagar and Janetzke, 1993) because it also spawned jobs to execute stand-alone computer programs.

After the stand-alone program completes its execution, control returns to the consequence module, which resumes execution. Normally the module then reads an output file created by the stand-alone program. The output is read into the appropriate arrays and returned to the EXEC. This style of execution is currently used in modules such as EBSREL, UZFT, SZFT, and ASHPLUME (described in chapter 4). EBSREL executes the release portion of the EBSPAC code (Mohanty et al., 1996) and both UZFT and SZFT execute the NEFTRAN program (Olague et al., 1991). ASHPLUMO executes the ASHPLUME program (Jarzemba et al., 1997). This method is also employed in the SAMPLER utility module. When the option for using Latin Hypercube Sampling is selected, the program by Iman and Shortencarier (1984) is executed. A summary of the current implementation for consequence modules is provided in table 2-1. Some modules have attributes of more than one style of implementation. For example, the NFENV module uses analytical equations to calculate temperatures, relative humidity (RH), and groundwater flow rates. It uses table-lookup for the chemical composition of the groundwater. The predominant implementation is based on subroutines, as indicated in table 2-1.

In summary, the EXEC controls the execution and flow of data between consequence modules, and utility modules such as READER and SAMPLER (described in chapter 3) as well as the generation of output files. The output files can be used for parameter importance analyses, generation of time-dependent risk curves, and generation of complementary cumulative distribution functions for cumulative release of radionuclides and peak dose. Utility modules are used to read the input file, control the sampling and distribution of sampled parameters, ensure a consistent description of the repository system and provide generic computational algorithms that all consequence modules can access. The consequence modules embody the conceptual and mathematical models of the features, events, and processes deemed important for conducting a TSPA. The utility modules and consequence modules are described in more detail in the next two chapters.

**Table 2-1. Current implementation for consequence modules in the TPA Version 3.0 code**

<b>Consequence Module</b>	<b>Table-Look Up</b>	<b>Subroutine(s)</b>	<b>External Standalone Program</b>
UZFLOW		X	
NFENV		X	
EBSFAIL			X
EBSREL			X
UZFT			X
SZFT			X
DCAGW	X		
CLIMATO		X	
FAULTO		X	
SEISMO		X	
VOLCANO		X	
ASHPLUMO			X
ASHRMOVO		X	
DCAGS	X		

### 3 UTILITY MODULES

The TPA Version 3.0 code performs various initializations, input and output processing, and intermediate calculations. There are six primary utility modules consisting of: **READER**, **SAMPLER**, **MODULE VARIABLE**, **INVENT**, **SUBAREA**, and **ARRAY**. Each of these utility modules is composed of a variety of subroutines and function routines that provide centralized support to the algorithms in the consequence modules. Descriptions of these utility modules and their support functions are presented in this section. In appendix C, more detailed information is provided about specific functions and subroutines for each utility module.

#### 3.1 READER

**READER** is a utility module that preprocesses the data from the *tpa.inp* file and is the only subroutine that reads the *tpa.inp* file. The *tpa.inp* file contains data specific for the TPA code execution as well as all probability density function (PDF) definitions for parameters that will be provided to the consequence modules. Additional discussion on the specific input parameters and format requirements are provided in chapter 5.

One significant feature of **READER** is that it has a large number of error traps that detect problems with the input data and provide specific error messages directing the analyst to the problem. This is a feature that did not exist in the TPA Version 2.0 code. Based on experience exercising the earlier version of the code, it was considered highly desirable to have a **READER** with numerous built-in error traps. Examples of error traps include:

- Checking for two title lines
- Checking for blank title lines
- Checking for appropriate parameter ranges for PDFs
- Checking keywords such as aqueousnuclides, subarea, and title are not used more than once
- Checking that sufficient repository area is specified to dispose of 70,000 MTU

Most error traps also identify the line number of the *tpa.inp* file where the error was detected. Below are a number of errors and the error messages generated by **READER**.

**READER** will enter an error trap if the user has only one title line in the *tpa.inp* file. Below is an example of a title being defined, but the second title line is missing:

```
**
title
  Only one title line supplied
**
```

The asterisks (\*\*) are comment lines that help the user visually block the input data into groups. The error generated by **READER** is as follows:

```
***>>> Error in Reader <<<***
  Second line of title is blank or comment line
  title line is:
**
  Look on Line Number =    4
```

The error message prints the second line of the title (which is a comment line) and the line number of the *tpa.inp* file where the error was detected.

Another error trap example is when a flag value is set to a value other than zero or one:

```
**
iflag
Seismic disruptive scenario flag (yes=1, no=0)
5
**
```

The resulting error message is:

```
***>>> Error in Reader <<<***
(iflag .ne. 0) .and. (iflag .ne. 1)
iflag = 5
Look on Line Number = 16
```

Another example is when the user specifies that zero time steps are to be used:

```
**
iconstant
NumberOfTimeSteps
0
**
```

then the following error message is provided by READER:

```
***>>> Error in Reader <<<***
NumberOfTimeSteps .le. 1
needs to be >= 2
```

The last example is when the user specifies a uniform distribution but the lower limit is greater than the upper limit:

```
**
uniform
GroundwaterPercolationRate[mm/yr]
2.5, 2.0
**
```

READER generates the following error message:

```
***>>> Error in Reader <<<***
for uniform distribution
name = GroundwaterPercolationRate[mm/yr]
xmax .le. xmin
xmax = 2.000000000000000
xmin = 2.500000000000000
Look on Line Number = 176
```

The description of keywords recognized by READER are described in chapter 5. The above examples only highlight that READER has many error traps to assist the analyst in developing an acceptable input file for the TPA Version 3.0 code.

## 3.2 SAMPLER

The SAMPLER utility module dynamically stores and retrieves information for model input parameters with assigned statistical distributions. PDFs are read from the *tpa.inp* file during program execution. SAMPLER also supports either Monte Carlo sampling or Latin Hypercube Sampling (LHS). The user selects the sampling scheme in the *tpa.inp* file. If Monte Carlo is selected, then samples are drawn from the various distributions using well established algorithms (e.g., Press et al., 1986; Ripley, 1987; Ang and Tang, 1984). If LHS is selected, then the program by Iman and Shortencarier (1984) is used. The statistical distributions included in the SAMPLER utility module are:

- Constant
- Uniform
- Loguniform
- Normal
- Lognormal
- Triangular
- Logtriangular (currently not supported when using LHS)
- Beta (currently not supported when using LHS)
- Logbeta (currently not supported when using LHS)
- Exponential (currently not supported when using LHS)
- Finite exponential (currently not supported when using LHS)

In addition, two other options exist which allow sampling from user supplied data and forcing a correlation between parameters.

- User data (currently not supported when using LHS)
- Correlate inputs

These distributions are discussed in the following sections.

### 3.2.1 Constant

A parameter may be specified as a constant such that it remains at a fixed value for the simulation. The constant can be described as a PDF consisting of a Dirac delta function located at the value of the constant [see Boyce and DiPrima (1977) for a discussion of the Dirac delta function]. However, this mathematical complexity is not needed for this discussion.

### 3.2.2 Uniform Distribution

The PDF for the uniform distribution is:

$$f(x) = \frac{1}{(B - A)}, \quad A < x < B \quad (3-1)$$

The mean of the uniform distribution is:

$$\mu = \frac{(A + B)}{2} \quad (3-2)$$

and the variance is defined as:

$$\sigma^2 = \frac{(B - A)^2}{12} \quad (3-3)$$

To assign the uniform distribution to an input parameter, the user must specify the endpoints A and B. An example is shown in figure 3-1 where the end points are A=1.2 and B=2.45.

### 3.2.3 Loguniform Distribution

The loguniform distribution is a variation of the uniform distribution, and is built upon calling the uniform distribution. The actual end points of the loguniform distribution are (a,b). The endpoints passed to the uniform distribution are the log transformation of the actual end points, [e.g., A=log(a), B=log(b)]. The exponential of the sampled value returned from the uniform distribution, is then returned as the value from the loguniform distribution.

### 3.2.4 Normal Distribution

The PDF of the normal distribution is:

$$f(x) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left[-\frac{(x - \mu)^2}{2\sigma^2}\right], \quad -\infty < x < \infty \quad (3-4)$$

Where  $\mu$  and  $\sigma$  are the mean and standard deviation of the distribution, respectively. SAMPLER generates samples having standard normal distribution (mean of 0.0 and standard deviation of 1.0) with lower and upper cut-off A and B. Sampling from the normal PDF is based on the algorithm described by Press et al. (1986) when Monte Carlo sampling is selected, and when LHS is specified the algorithm in the code by Iman and Shortencarier (1984) is used. SAMPLER requires specification of A and B at the 0.001 and 0.999 quantiles of the normal distribution:

$$P(x < A) = 0.001 \quad (3-5)$$

and

$$P(x > B) = 0.999 \quad (3-6)$$

The mean of the normal distribution is:

$$\mu = \frac{A + B}{2} \quad (3-7)$$

and the variance of the normal distribution is:

$$\sigma^2 \approx \left( \frac{B - A}{6.18} \right)^2 \quad (3-8)$$

An example of a normal distribution with a mean of 2.0 and standard deviation of 3.5 is shown in figure 3-2. With this mean and standard deviation the 0.001 and 0.999 quantiles are  $A=-8.816$  and  $B=12.816$ , respectively.

### 3.2.5 Lognormal Distribution

The lognormal distribution is built upon using the normal distribution. Here, the logs of the 0.001 and 0.999 quantiles [e.g.,  $A=\log(a)$ ,  $B=\log(b)$ ] are used as inputs to sample from the normal distribution. The exponential of this value is returned as the final sampled parameter value.

### 3.2.6 Triangular Distribution

The triangular distribution is described by a minimum value (A), maximum value (C) and a mode or peak value (B). The PDF of the triangular distribution is:

$$\begin{aligned} f(x) &= \frac{2(x - A)}{(C - A)(B - A)}, & A < x < B \\ &= \frac{2(C - x)}{(C - A)(C - B)}, & B < x < C \end{aligned} \quad (3-9)$$

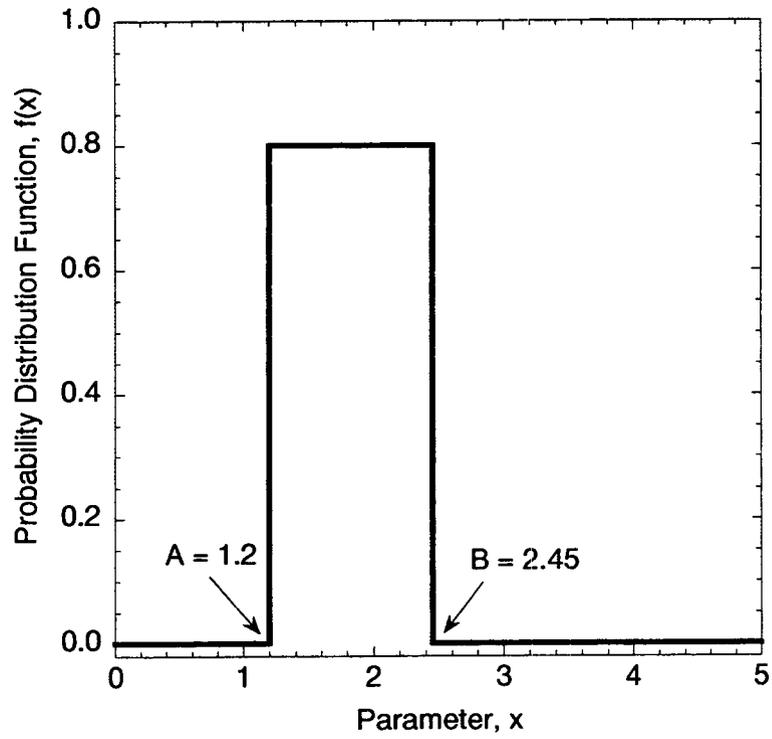
The mean of the triangular distribution is:

$$\mu = \frac{A + B + C}{3} \quad (3-10)$$

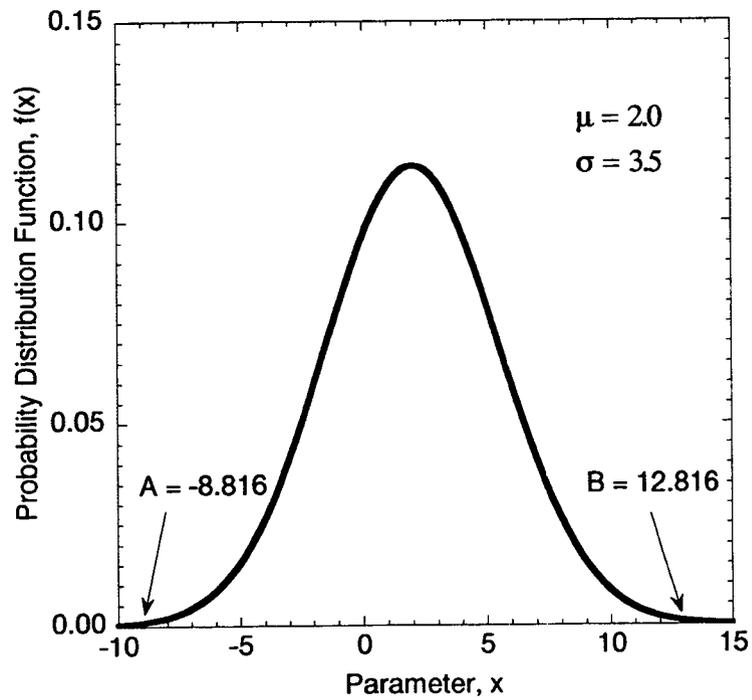
and the variance is defined as:

$$\sigma^2 = \frac{A(A - B) + B(B - C) + C(C - A)}{18} \quad (3-11)$$

An example triangular distribution is shown in figure 3-3. Here the lower limit, the location of the peak, and upper the limit are  $A=-2$ ,  $B=4$ , and  $C=14$ , respectively.



**Figure 3-1. Example of uniform distribution**



**Figure 3-2. Example of normal distribution**

### 3.2.7 Logtriangular Distribution

The logtriangular distribution is based on using the triangular distribution. The minimum, peak, and maximum locations (a,b,c) are transformed by taking their logs [e.g.,  $A=\log(a)$ ,  $B=\log(b)$ , and  $C=\log(c)$ ]. The sample is drawn from the triangular distribution. The exponential of this sample is then returned from the logtriangular distribution.

### 3.2.8 Beta Distribution

The PDF of the beta distribution within 0 and 1 (standard beta distribution) is:

$$f(y) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} y^{\alpha-1} (1 - y)^{\beta-1}, \quad 0 < y < 1 \quad (3-12)$$

where  $\alpha$  and  $\beta$  are positive, shape parameters of the distribution and  $\Gamma(\eta)$  is the Gamma function which is defined as:

$$\Gamma(\eta) = \int_0^{\infty} z^{\eta-1} e^{-z} dz \quad (3-13)$$

where  $z$  is a dummy variable. The mean and variance of the standard beta distribution are:

$$\mu = \frac{\alpha}{\alpha + \beta} \quad (3-14)$$

and

$$\sigma^2 = \frac{\alpha\beta}{(\alpha + \beta)^2 (\alpha + \beta + 1)} \quad (3-15)$$

The general beta distribution with lower and upper bounds  $A$  and  $B$  (i.e.,  $A < x < B$ ) may be obtained from mapping the standard beta distribution:

$$x = A + y (B - A) \quad (3-16)$$

where  $y$  corresponds to the sample from the standard beta distribution. When Monte Carlo sampling is selected, the algorithm by Ang and Tang (1984) is employed. When LHS sampling is selected, the beta distribution can not be used because it is not currently working in the code by Iman and Shortencarier (1984). An example of a beta distribution is shown in figure 3-4 where the shape parameters are  $\alpha=2$  and  $\beta=6$ , and the lower and upper limits are  $A=-3$  and  $B=6$ , respectively.

### 3.2.9 Logbeta Distribution

The logbeta distribution uses the beta distribution. The upper and lower bounds are transformed by taking the logarithm [e.g.,  $A=\log(a)$ ,  $B=\log(b)$ ]. The sample is then drawn from the beta, and the exponential of the sample is returned from the logbeta.

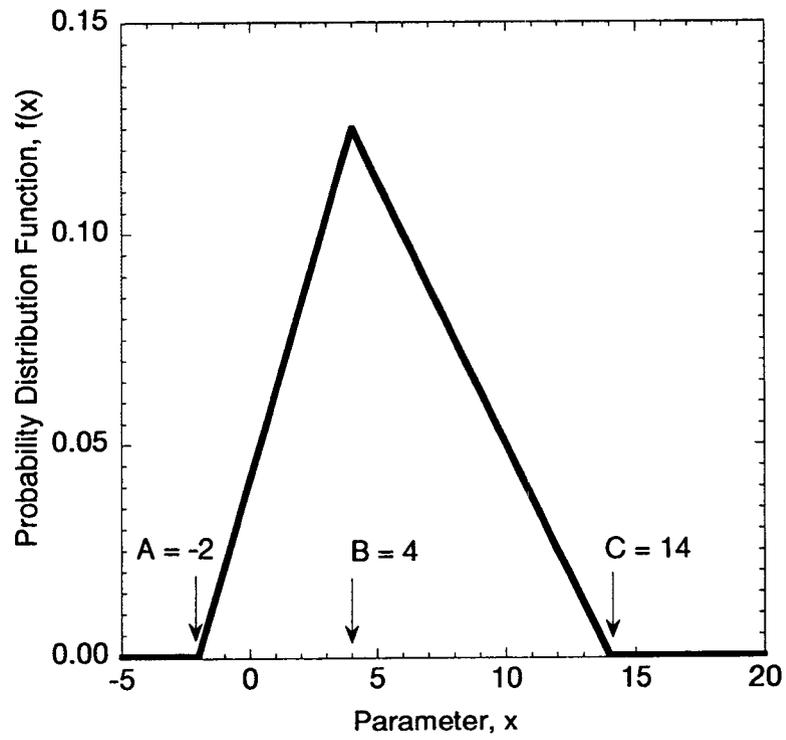


Figure 3-3. Example of triangular distribution

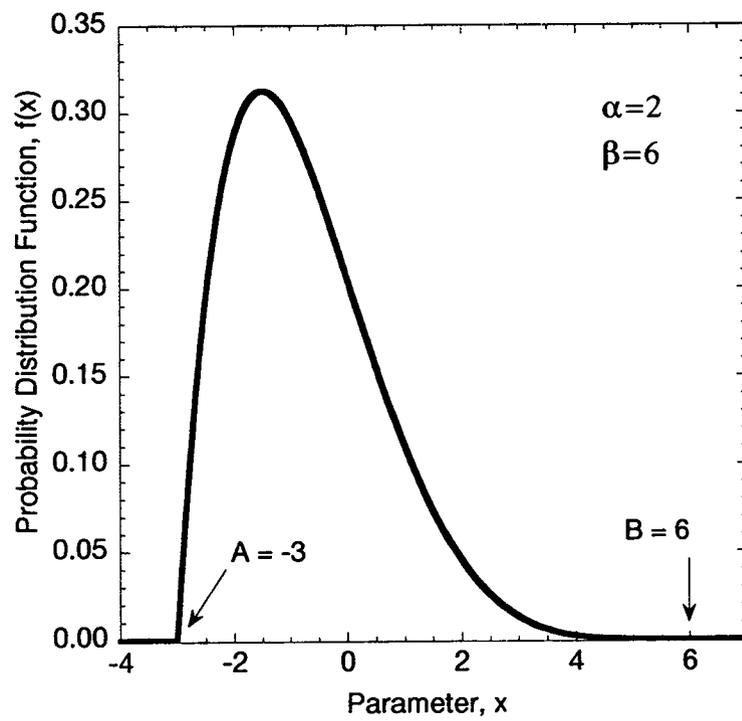


Figure 3-4. Example of beta distribution

### 3.2.10 Exponential Distribution

The PDF of an exponential distribution is:

$$f(x) = \lambda e^{-\lambda x}, \quad 0 < x \quad (3-17)$$

where  $\lambda$  is the recurrence probability. The exponential distribution frequently is used to describe the time of the next event such as faulting, volcanism, or seismic activity. In these cases,  $\lambda$  is the annual recurrence probability which is the reciprocal of the return period. The mean of the exponential distribution is:

$$\mu = \frac{1}{\lambda} \quad (3-18)$$

and the variance is:

$$\sigma^2 = \left(\frac{1}{\lambda}\right)^2 \quad (3-19)$$

An example of an exponential distribution is shown in figure 3-5. Here, the recurrence probability is equal to 0.4. The figure also illustrates that  $x$  is restricted to be positive.

### 3.2.11 Finite Exponential Distribution

The PDF for the finite exponential distribution:

$$f(x) = \left( \frac{\lambda}{e^{-\lambda A} - e^{-\lambda B}} \right) e^{-\lambda x} \quad 0 \leq A < x < B \quad (3-20)$$

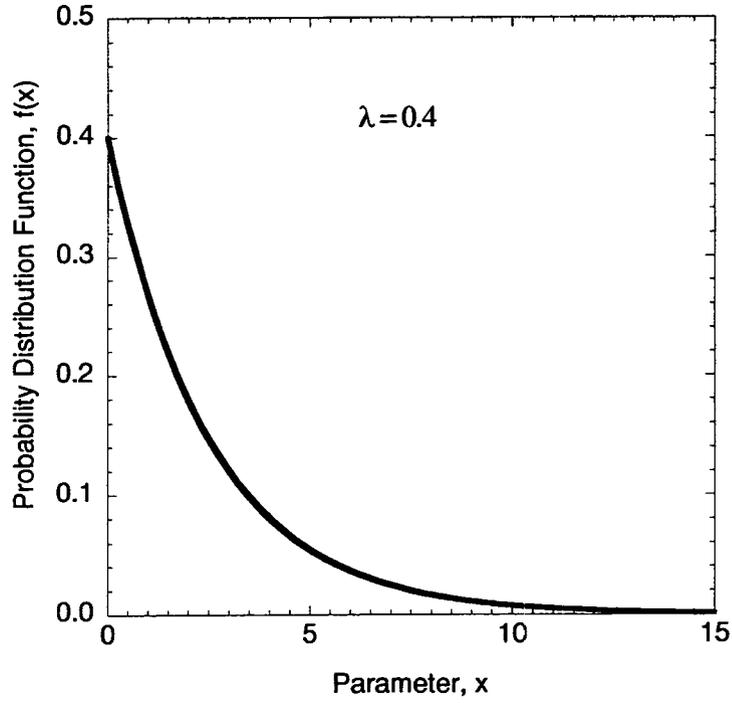
The PDF is equal to zero outside the range from  $A$  to  $B$ . When the return period is long compared to  $(B-A)$ , the finite exponential is approximately a uniform distribution.

An example of a finite exponential distribution is shown in figure 3-6. Here the recurrence probability is 0.4, and the lower and upper limits are  $A=2$  and  $B=5$ , respectively.

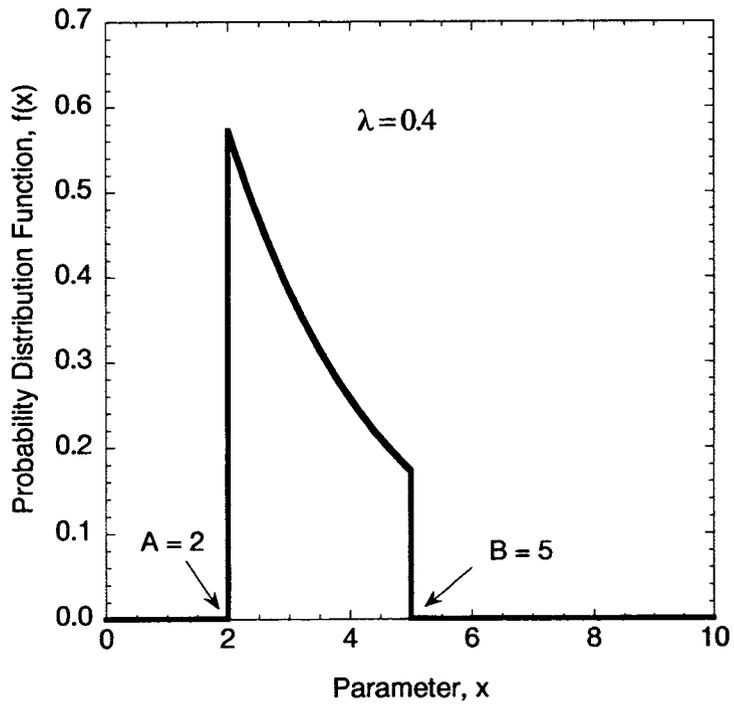
### 3.2.12 User-Defined Distribution

SAMPLER also allows the input of user-supplied data for sampling an empirical distribution. For instance, suppose the information for an input variable is only available in the form of sample data. The user-defined distribution option provides the means to accommodate this data and sample from it. This is accomplished by sampling directly from the empirical distribution function formed from available data.

For example, suppose that there are eight equally probable sample data points: 0.3, 0.7, 1.3, 1.8, 2.0, 2.6, 3.0, 3.3. The cumulative distribution function (CDF) for these example data is shown in figure 3-7. The step heights (probabilities) are all the same and in this case equal 1/8. Samples from this distribution will be one of the eight input values, and neither interpolation nor extrapolation of the data is performed. Hence, the CDF has a stair-step profile.



**Figure 3-5. Example of exponential distribution**



**Figure 3-6. Example of finite exponential distribution**

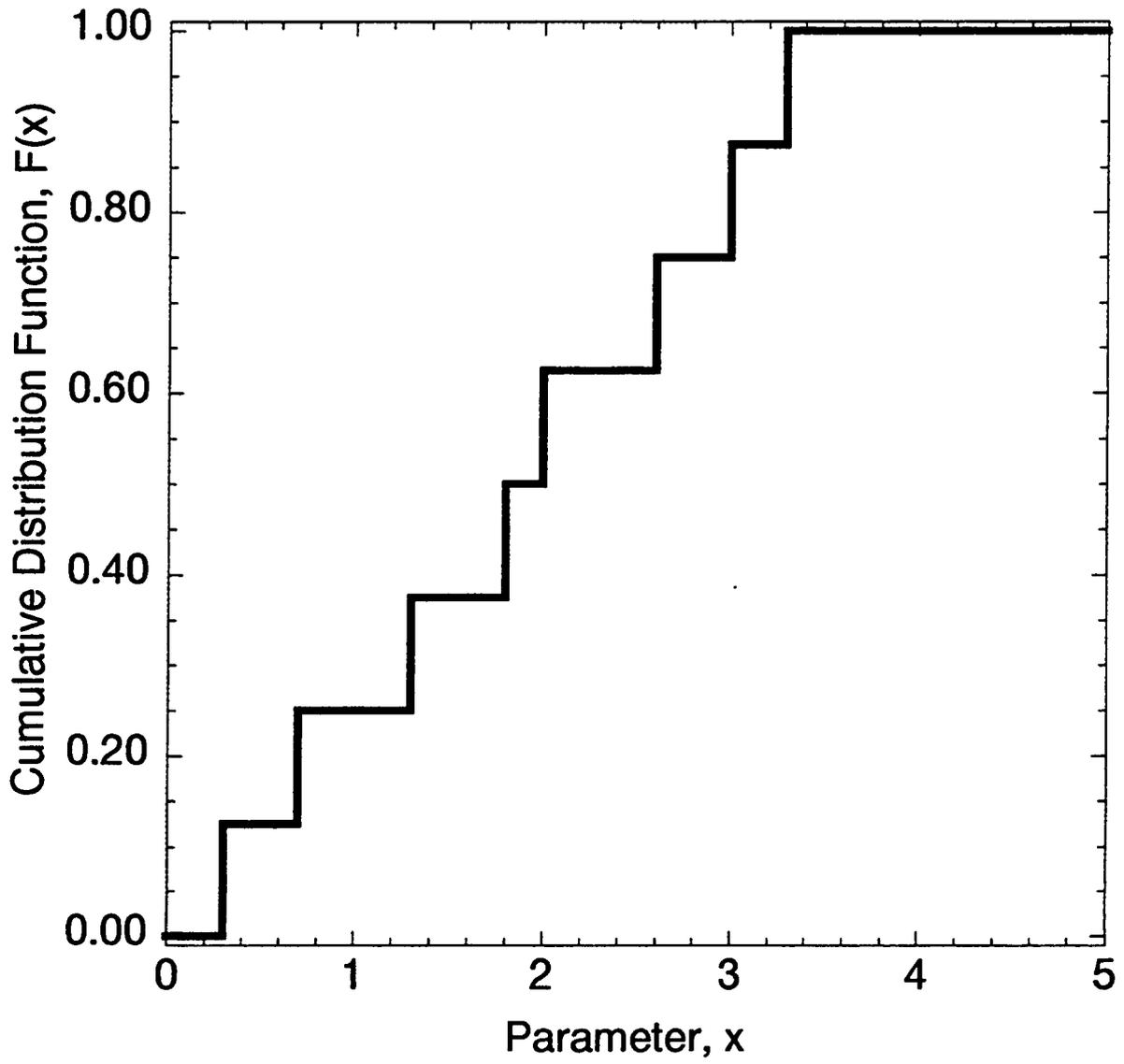


Figure 3-7. Example of user-defined distribution

### 3.2.13 Specified Correlation

SAMPLER utility also contains a correlation feature that samples two variables to introduce a user specified correlation. The correlation coefficient ( $\rho$ ) is specified for the rank transformation of the inputs. When  $\rho = 0$ , the two variables are independent. When  $\rho = 1$ , there is a perfect positive correlation and when  $\rho = -1$ , there is a perfect negative correlation. The algorithm for correlating inputs is described on page 321 of Benjamin and Cornell (1970). The algorithm was initially established for two normal (or Gaussian) distributions but has been generalized in SAMPLER for any two distributions. The algorithm begins by assuming the first parameter is independent and the second parameter is sampled to enforce the correlation. A random sample is taken from the standard normal distribution (e.g., mean=0, standard deviation =1).

$$z_1 = N(0,1) \quad (3-21)$$

The value of  $z_1$  influences the distribution of the second variable, such that:

$$z_2 = N(\rho z_1, \sqrt{1-\rho^2}) \quad (3-22)$$

The mean of the second distribution is equal to the product of the correlation coefficient ( $\rho$ ) and the value of the first sample ( $z_1$ ). The standard deviation of the second example is a function of the correlation coefficient. When  $\rho=0$ , the second distribution is the same as the first. When  $|\rho|=1$ , then the standard deviation of the second distribution is very small.

These results can be generalized to any distribution. First, the  $z$ 's from the normal distribution are transformed to  $u$ 's for a uniform distribution. These  $u$ 's are the quantiles of the parameter distributions. This transformation is performed by computing the CDF of the normal distribution using  $z_1$  and  $z_2$ . This yields:

$$u_1 = F(z_1) \quad (3-23)$$

$$u_2 = F(z_2) \quad (3-24)$$

where  $F(z)$  is the CDF for the standard normal distribution,  $N(0,1)$ . The  $u$ 's are then used to invert the CDF of the respective distributions to obtain parameter values.

$$x_1 = F_1^{-1}(u_1) \quad (3-25)$$

$$x_2 = F_2^{-1}(u_2) \quad (3-26)$$

In general, the CDFs of each parameter are unique, hence the subscript has been used on the  $F$ 's. When using the LHS module option, an alternative algorithm has been implemented by Iman and Shortencarier (1984). The alternative algorithm is not described here. Because it is much more complex to implement a matrix of correlations between multiple parameters and because the code by Iman and Shortencarier handles this case, the analyst should select the LHS scheme when a matrix of correlations is specified.

An example of a user specified correlation is shown in figure 3-8. On the left side, scatter plots of the z's, u's, and x's are shown between two parameters when  $\rho=0.0$  (i.e., no correlation). On the right side, similar scatter plots are shown for  $\rho=0.9$  (parameters highly correlated with positive correlation). These plots visually demonstrate how the algorithm correlates the z's (samples from unit normal distributions), transforms them into u's (quantiles), and then transforms these into x's (sampled values). The first parameter has a normal distribution which is the same as shown in figure 3-2. The second parameter has a beta distribution which is the same as shown in figure 3-4.

### 3.3 MODULE VARIABLE

The MODULE VARIABLE utility module consists of subroutines for storing values computed by consequence modules. MODULE VARIABLE provides a database for storing consequence module results (i.e., subsystem and system performance measures) and provides a special index to identify each parameter value. This procedure provides data security, such that only the routine that introduced the data knows the index for a given parameter and may change its value within the database. Other modules within the TPA Version 3.0 code are allowed to query the value; however, they are unable to change the value in the database.

MODULE VARIABLE provides the analyst using the TPA Version 3.0 code with a tool for storing parameter values for later correlation with the output. For example, the analyst may decide to save the time for each faulting event to assess the sensitivity of the resulting dose to the event time. Examples of the types of variable information generally saved to the MODULE VARIABLE utility module include:

- WP Failure Time (yr)
- Fractional Release Rate (1/yr)
- Time of Peak Annual Dose (yr)
- Magnitude of Peak Annual Dose (rem/yr)
- Cumulative Normalized Release

### 3.4 INVENT

INVENT is a utility module that centralizes the computation and storage of radionuclide inventory data for use throughout the TPA Version 3.0 code. This module builds upon and extends the work of Lozano et al. (1994). The subroutines provide the inventory (in Ci/MTU) of 43 radionuclides for times up to and beyond one million years, accounting for chain decay and ingrowth of daughters. Information for half-life, specific activity, and the EPA release limit are also available for each radionuclide. Thermal output of the average HLW (MTU weighted average of BWR and PWR spent nuclear fuel) is also provided by the INVENT module.

During the execution of the TPA Version 3.0 code, radionuclide inventories must repeatedly be calculated and queried. For example, in the IPA Phase 2 exercise (Wescott et al., 1995) the inventories of 20 radionuclides were tracked from 10 yr out-of-core (assumed age at emplacement) up to 10,000 yr. From experience with the TPA Version 2.0 code, computation of the nuclide inventories were centralized to ensure data consistency between modules, allow for the variability of the initial nuclide inventories due to prolonged aging of the waste from reactor to repository, and possibly change the composition of the waste. The INVENT utility module was developed to centralize the calculation of radionuclide inventories for the spent nuclear fuel and to create an efficient storage and retrieval interface for other modules to access this information.

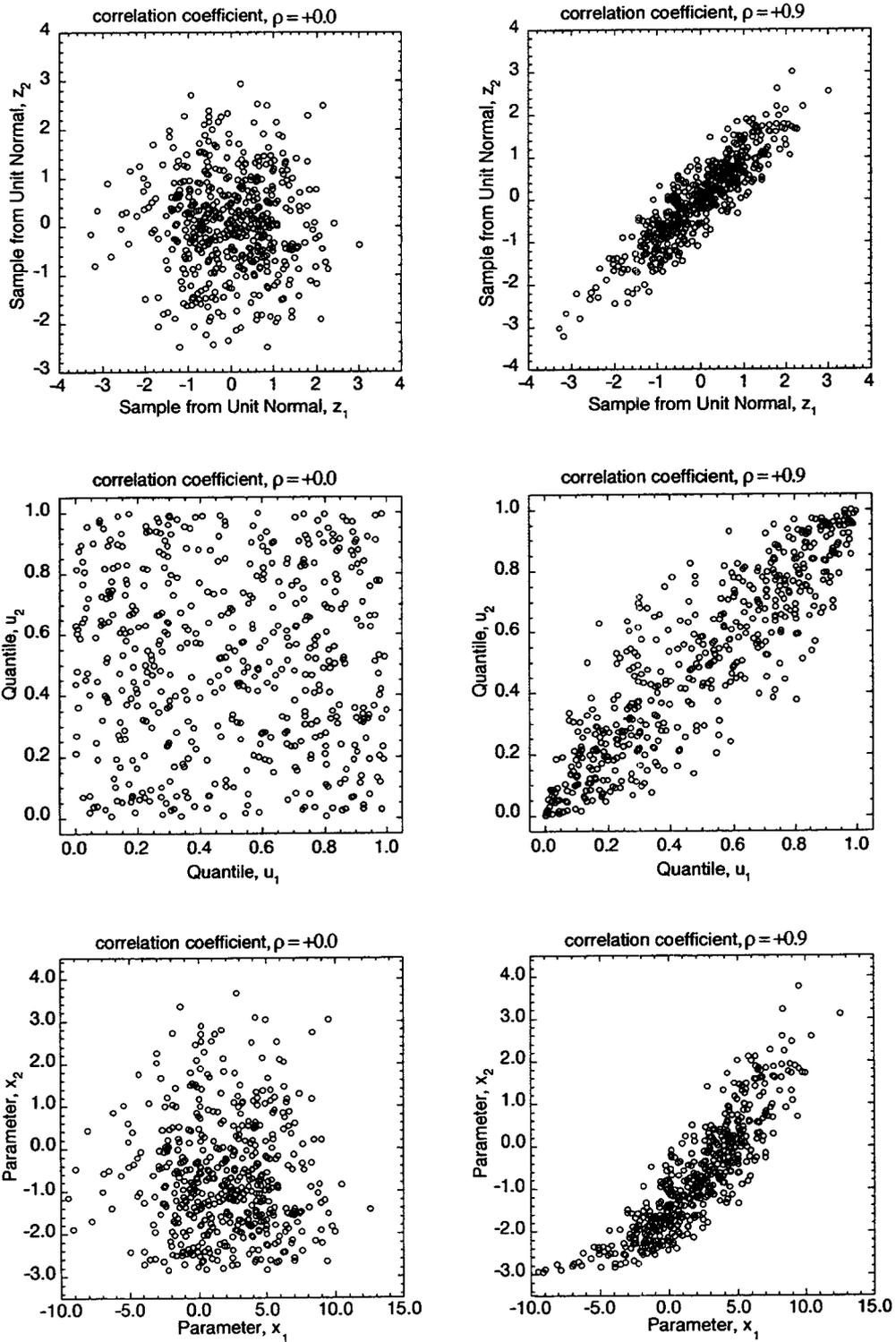


Figure 3-8. Example of specified correlation between two sampled parameters

The governing ordinary differential equation for predicting radionuclide inventories as a function of time is:

$$\frac{d}{dt}N_i(t) = \lambda_{i-1}^P N_{i-1}(t) - \lambda_i^T N_i(t) - R_i \quad (3-27)$$

where

$N_{i-1}(t)$  = the parent radionuclide population as a function of time [mol]

$N_i(t)$  = the radionuclide population as a function of time [mol]

$\lambda_{i-1}^P$  = the physical (radioactive) decay constant for the parent radionuclide [1/yr]

$\lambda_i^P$  = the physical (radioactive) decay constant for the radionuclide [1/yr]

$\lambda_i^T$  = the total removal constant, which is a combination of physical decay and any other removal processes whose rate of removal is proportional to the population of the nuclide

The analytic solution to the above differential equation is:

$$N_i(t) = \sum_{j=1}^i \left[ \frac{\sum_{n=1}^j \left[ \prod_{m=1}^{i-1} (\text{TERM}_m) \right] (N_{n,0} + R_n/\lambda_j^T)}{\prod_{\substack{k=j \\ k \neq j}}^i (\lambda_k^T - \lambda_j^T)} \exp(-\lambda_j^T t) \right] - \frac{\sum_{u=1}^i \left[ \prod_{s=1}^{i-1} (\text{TERM}_s) \right] R_u}{\prod_{q=1}^i \lambda_q^T} \quad (3-28)$$

where

$$\text{TERM}_m = \lambda_{n-m}^T - \lambda_j^T \quad m < n$$

$$\lambda_m^P \quad m \geq n$$

$$\text{TERM}_s = \lambda_s^T \quad s < u$$

$$\lambda_s^P \quad s \geq u$$

$N_{n,0}$  = the initial population of radionuclide

In the current implementation of the INVENT utility module, the  $R_i$  terms are universally equal to zero and  $\lambda_i^T$  is equal to  $\lambda_i^P$  for all radionuclides. The above solution was originally derived for calculating radionuclide leaching from contaminated volcanic ash blankets. The full derivation of these equations will be the subject of a future paper on this topic.

The initial inventories in the INVENT utility module are based on calculations using the ORIGEN2 code (Croff, 1983; Ludwig and Renier, 1989). The INVENT module waste inventories are based on the default assumptions of an average of 65 percent PWR waste with a 42 GWd/MTU burnup

yr (i.e., default value), unless specified otherwise in the *tpa.inp* file. INVENT uses this information to determine the activity of each of the 43 radionuclides of interest in the HLW projected to the time of interest. The set of 43 radionuclides was selected after reviewing the literature and finding 43 is the largest set of radionuclides being considered in other TSPA efforts (Barnard et al., 1992; Wilson et al., 1994; TRW Environmental Safety Systems, Inc., 1995). The NRC IPA Phase 2 effort tracked 20 radionuclides, hence the database has been expanded.

The 43 radionuclides considered in the INVENT module contain four major actinide element decay chains. These chains are shown in figures 3-9, 3-10, 3-11, and 3-12. The specific nuclides being tracked are boxed in the decay chains. Additional radionuclides are tracked in the INVENT module which are not chain decay members, and are handled through simple decay equations. These radionuclides are typically fission or activation products and include:  $^{232}\text{U}$ ,  $^{151}\text{Sm}$ ,  $^{137}\text{Cs}$ ,  $^{135}\text{Cs}$ ,  $^{129}\text{I}$ ,  $^{126}\text{Sn}$ ,  $^{121\text{m}}\text{Sn}$ ,  $^{108\text{m}}\text{Ag}$ ,  $^{107}\text{Pd}$ ,  $^{99}\text{Tc}$ ,  $^{93}\text{Mo}$ ,  $^{94}\text{Nb}$ ,  $^{93}\text{Zr}$ ,  $^{90}\text{Sr}$ ,  $^{79}\text{Se}$ ,  $^{63}\text{Ni}$ ,  $^{59}\text{Ni}$ ,  $^{36}\text{Cl}$ , and  $^{14}\text{C}$ .

Radionuclide inventory calculations performed by INVENT have been verified by comparing the results with other published results, and plotting the inventories for times from 10 to 1,000,000 yr (figures 3-13a, 3-13b, 3-13c, and 3-13d). The trends in the inventories were compared with those published elsewhere (e.g., Roxburgh, 1987). As expected, most radionuclide inventories decrease with increasing time, some remain relatively constant over long periods of time (those with long half-lives), and others increase with time (daughters in a decay chain). For example,  $^{238}\text{Pu}$  has an 87.7 yr half-life, and its inventory can be observed to continuously decrease with time. Another example is  $^{234}\text{U}$ , which has a 244,500 yr half-life, and remains relatively constant up to about 100,000 yr. An example of daughter ingrowth can be seen with  $^{230}\text{Th}$ ,  $^{226}\text{Ra}$ , and  $^{210}\text{Pb}$ , which are in the  $^{246}\text{Cm}$  decay series of radionuclides (see figure 3-12). Hence, the inventories of these daughters increase with time. In summary, the INVENT module was developed to facilitate the calculation of radionuclide inventories and thermal output of the HLW which are used by various consequence modules of the TPA Version 3.0 code.

### 3.5 SUBAREA

SUBAREA is a utility module that performs storage and retrieval of repository subarea information. The data are read in the READER utility module, and stored for use by all modules. The consequence modules can acquire information about the subarea discretization, but are not allowed to change (i.e., inadvertently corrupt) the information.

The subareas are defined in the *tpa.inp* file by providing Universal Transverse Mercator location coordinates for the vertices of quadrilateral elements. The SUBAREA utility module has subroutines to query (or determine):

- Footprint area of subarea
- MTU of waste in subarea
- Number of WPs in subarea
- Subarea vertice coordinates
- Total number of subareas
- Coordinates of a subarea midpoint
- If a point is located within a specified subarea
- If a circle is located within a specified subarea
- If a line is located within a specified subarea

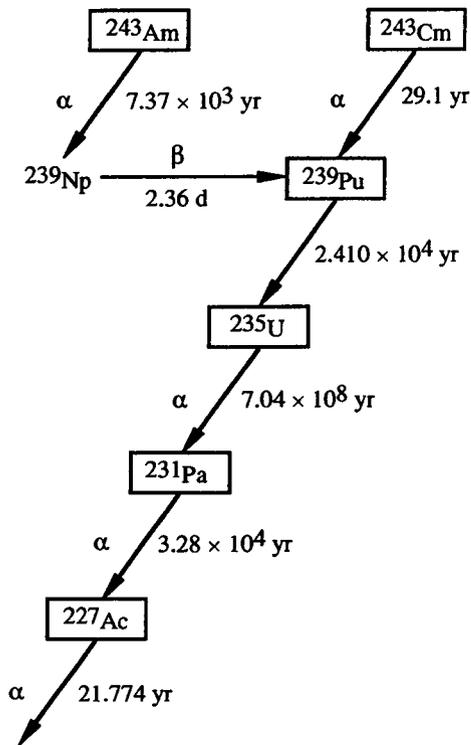


Figure 3-9. Cm-243 decay chain

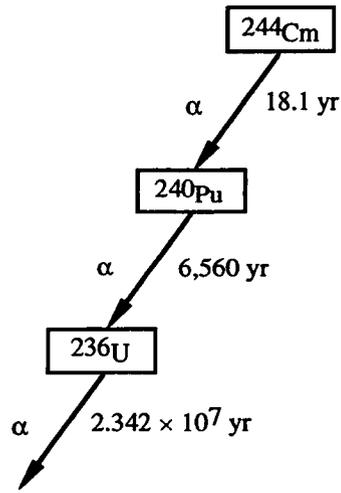


Figure 3-10. Cm-244 decay chain

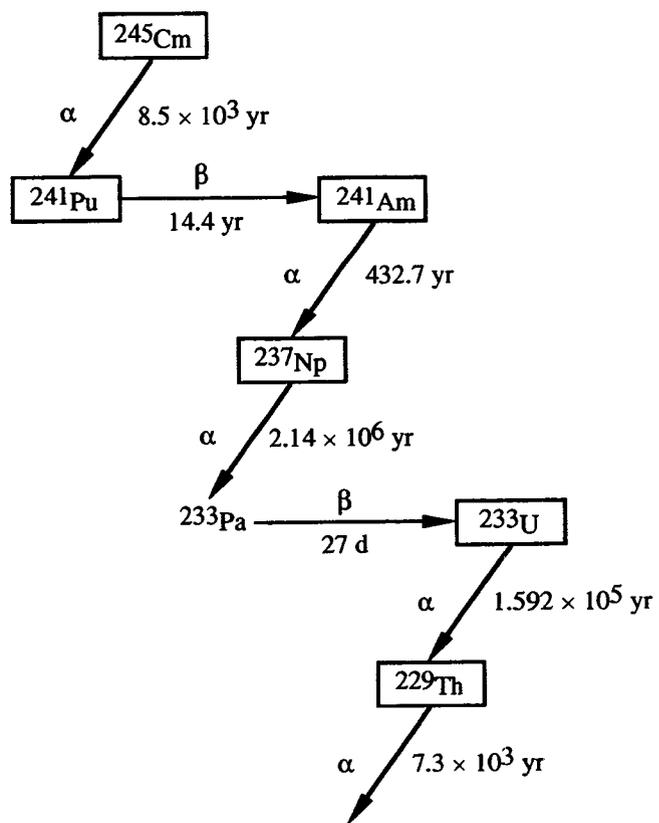


Figure 3-11. Cm-245 decay chain

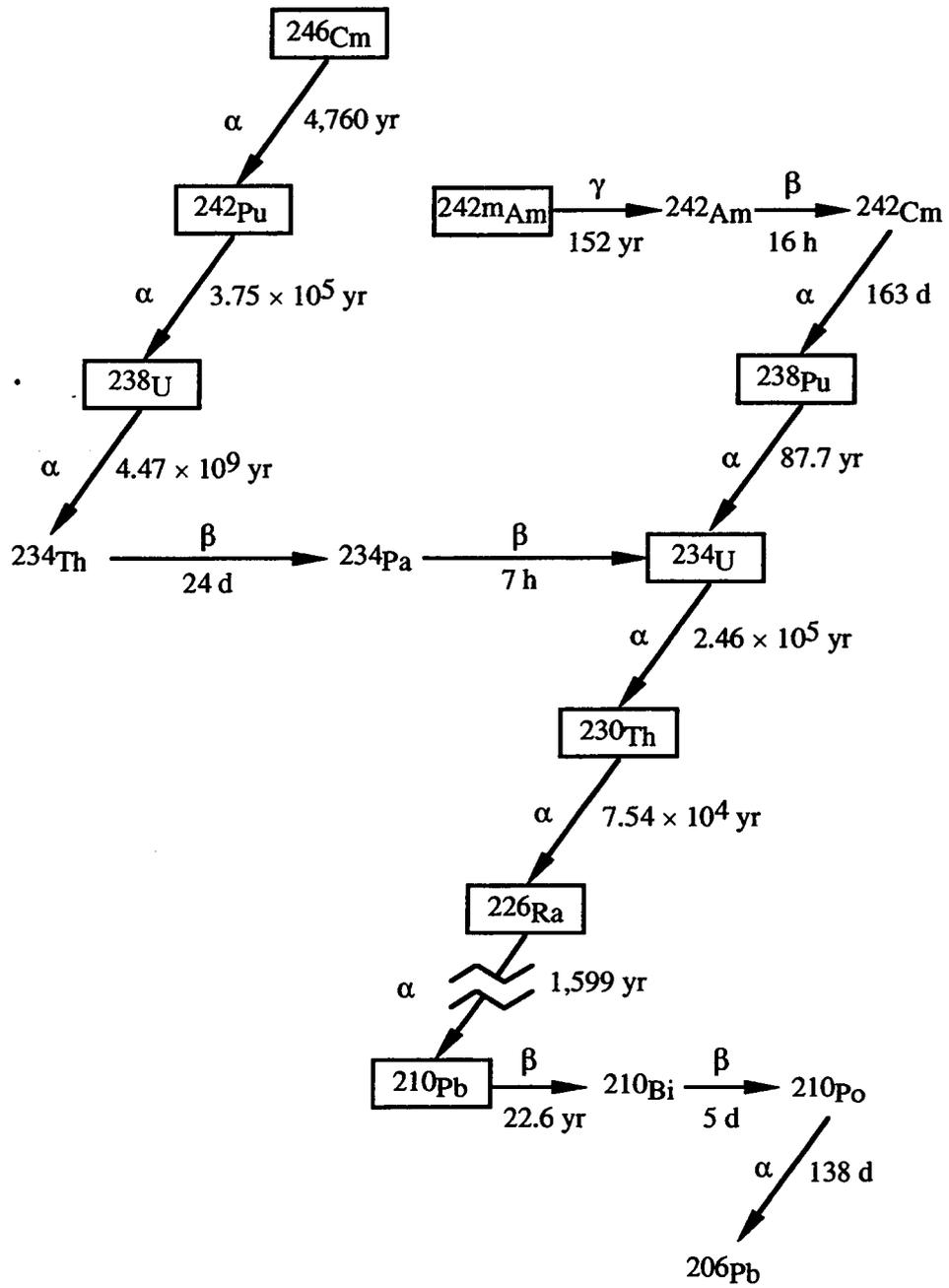


Figure 3-12. Cm-246 decay chain

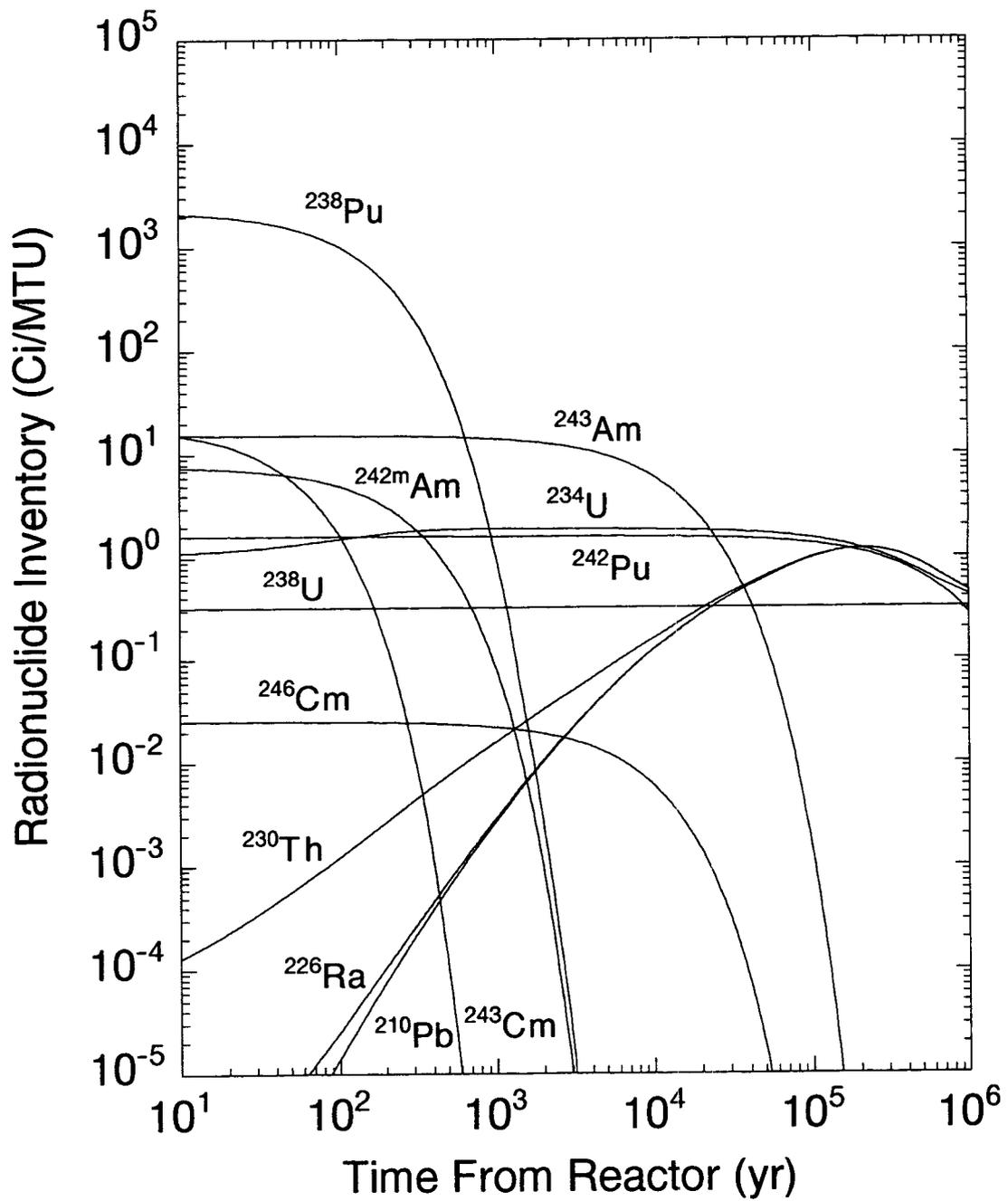


Figure 3-13a. Radionuclide inventories as a function of time (calculated by INVENT module)

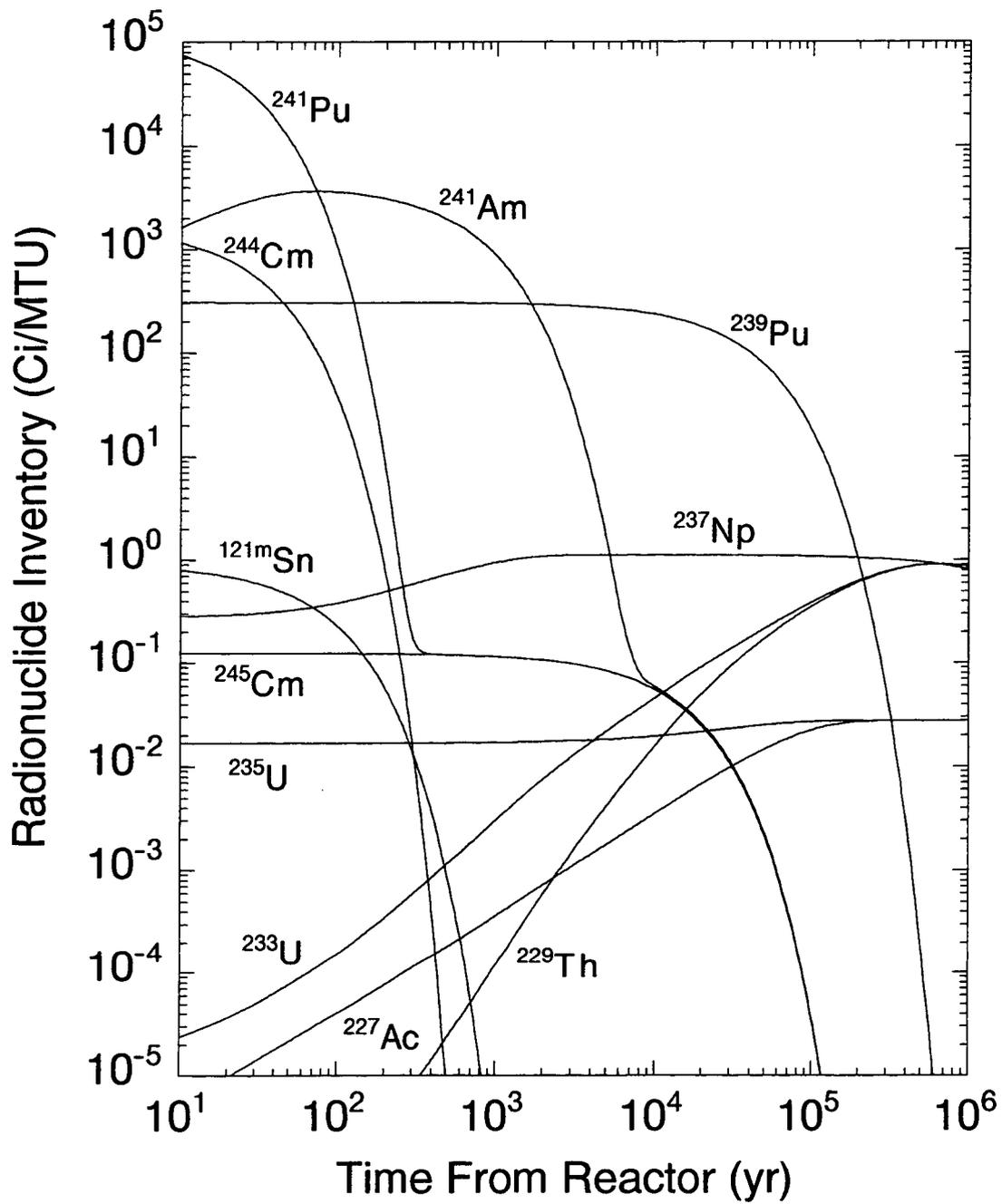


Figure 3-13b. Radionuclide inventories as a function of time (calculated by INVENT module)

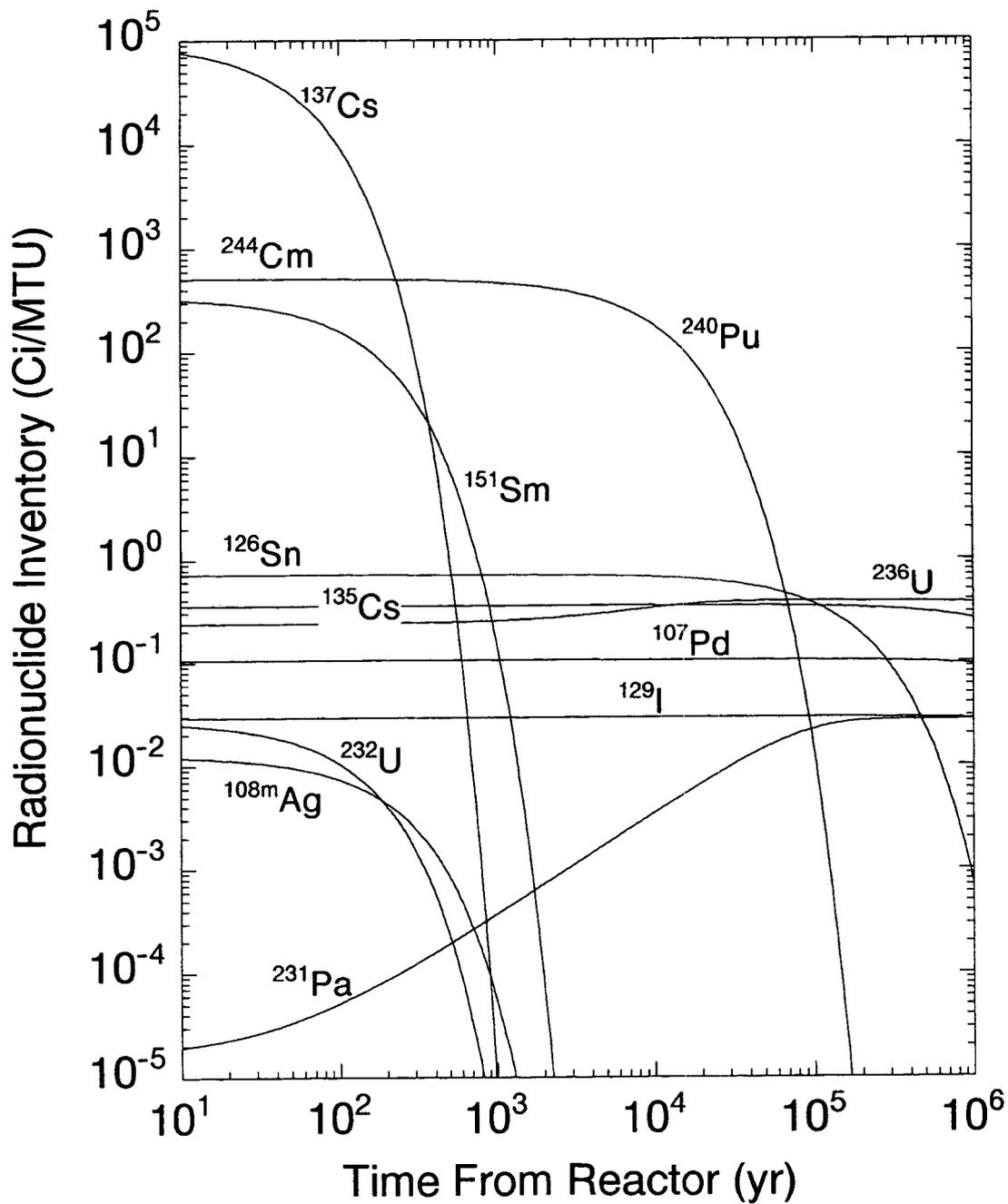


Figure 3-13c. Radionuclide inventories as a function of time (calculated by INVENT module)

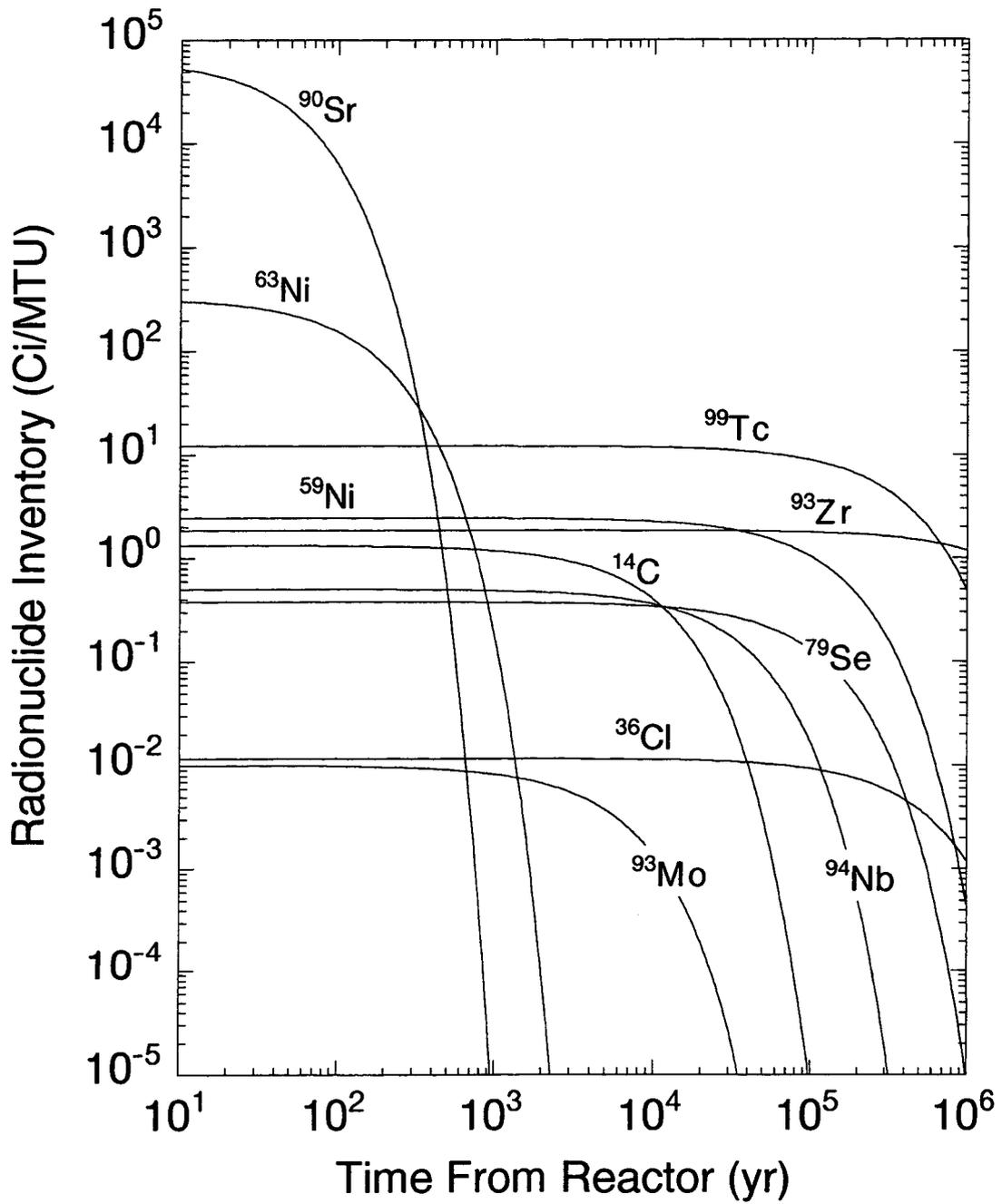


Figure 3-13d. Radionuclide inventories as a function of time (calculated by INVENT module)

For example, figure 3-14 shows four line segments which may represent faults or volcanic dikes that may intersect a given subarea. Subroutines in SUBAREA are currently being called in the FAULTO and VOLCANO modules for just such calculations. SUBAREA takes as input the coordinates of the end points of a given line segment and the vertices of a quadrilateral emplacement region. The routine then uses each segment of the boundary of the emplacement area, checks if there is a real intersection, and calculates the intersection point. SUBAREA then checks to determine if both ends of the line fall within the given region. If both ends of the line are within the given region, then the intersection length is the length of the line. If one end or both ends fall outside the given region, then SUBAREA determines the intersection point(s) of the line with 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. The intersection length for the first case is the distance between the real intersection point and the end point of the line that falls within the region. The intersected length for the second case is the distance between the two intersection points. In figure 3-14, for example, the intersection length of line 1 would be the full length, line 2 intersection length would be AB (need to find B only), and lines 3 and 4 the intersection lengths would be AB (need to find both A and B). SUBAREA calculates the intersection lengths of the lines in this example.

### 3.6 ARRAY

The ARRAY utility module contains 16 subroutines for the manipulation of various data types (i.e., floating point, integer, and character). These subroutines can be called by other modules to perform routine functions. The functionality of those subroutines range from ARRAY initialization, vector operations, variance calculations, index mapping, and sorting. The subroutines called by the ARRAY utility module are identified and described in table 3-1. In appendix C, more detailed information is provided for the specification of the subroutine calls.

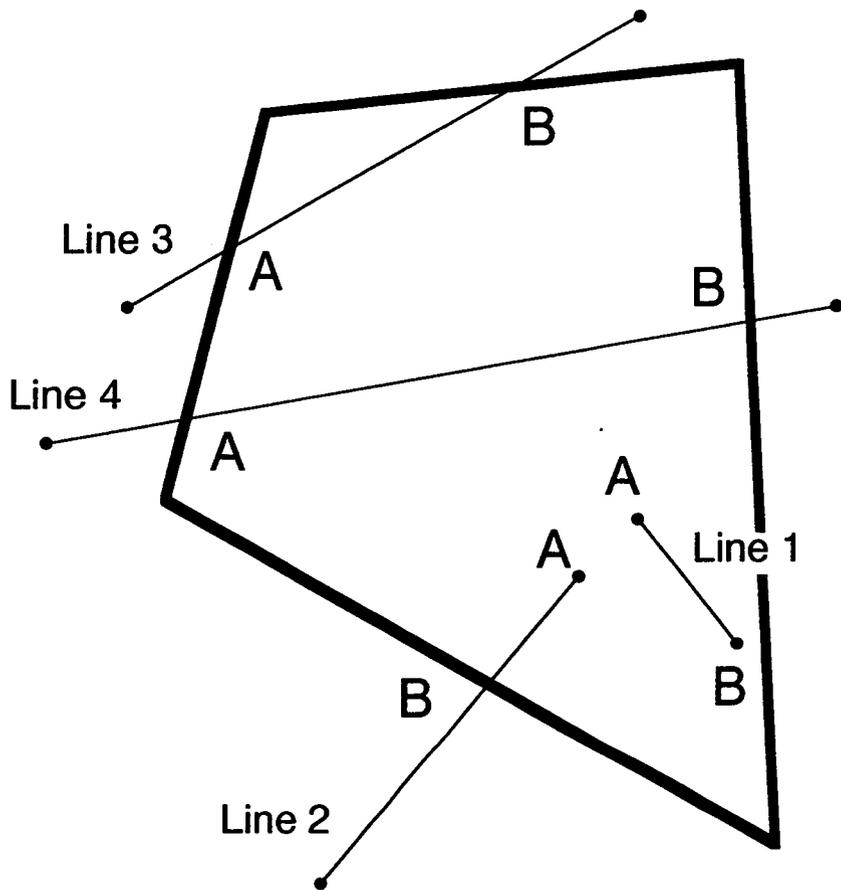


Figure 3-14. Subarea intersection example

**Table 3-1. Standard subroutines located in the ARRAY utility module**

Subroutine	Operational Description
zero	zeros out a vector
zeroi	zeros out integer vector
clearchar	clears character string
transpose	transposes the contents of a matrix
scale	scales a vector
scopy	scales and copies a vector
acopy	copies vector and adds a constant to each entry
ascopy	add, scale and copy vector
addto	adds one vector to another
isoneofset	determines if an integer is part of an integer set
ainterl	linearly interpolates in list of {time, value} data to find value at time of interest
avar	calculates variance of an array of values
amean	calculates mean of an array of values
checkinorder	determines if array of values are in order, either ascending or descending
sortqr	sorts based on pointers to array of values – sorts from smallest to largest value
maplist	maps data in first list into second list given the first set of {x,y} and the second set of {x}, and finds the second set of {y}

## 4 CONSEQUENCE MODULES

This chapter briefly describes each of the consequence modules associated with TPA Version 3.0 code. Each section contains a short explanation of the purpose of the module, the physical basis of the models encompassed by the module, and details on where to find more complete information on the given module. Two types of consequence modules are incorporated in the TPA Version 3.0 code: (i) flow, transport, and dose assessment modules and (ii) disruptive scenario modules. The former model releases from the repository to the CP, while the latter model disruptive scenarios. Some modules utilize both table-lookup and subroutines. A good example is NFENV. It uses semi-analytical solutions for temperatures and RH, but uses table-lookup for pH and chloride concentration based on MULTIFLO results.

### 4.1 UZFLOW

The unsaturated zone flow (UZFLOW) module, a set of TPA subroutines, calculates time-dependent unsaturated zone percolation flux into each subarea of the repository (see figure 4-1). UZFLOW uses a time history of mean annual precipitation (MAP) and mean annual temperature (MAT) generated by CLIMATO (the climate simulator described in section 4.8) to modify the mean annual infiltration (MAI) occurring under current and postulated future climatic conditions. Assuming that no lateral diversion occurs from the ground surface to the water table, and that the flow field in YM is in equilibrium, MAI can be equated to areally averaged deep percolation. This is a conservative formulation of deep percolation. It will be modified in the future to account for lateral flow. Accordingly, the areal average of MAI over a subarea is used for deep percolation. The calculated flux history is returned to EXEC which then passes this information to NFENV (the near-field environment module) and UZFT (the unsaturated zone flow and transport module).

UZFLOW discretizes each repository subarea into pixels (i.e., areas of 30 by 30 m). A digital elevation model (DEM) is used to assign elevation and soil depth to each pixel. Based on elevation, soil depth, soil and bedrock properties, and climatic variables, the MAI is estimated for each pixel, using an empirical relationship appropriate to YM (Stothoff et al., 1997). The empirical relationship was derived by analyzing the MAI generated from nearly 200 1D bare-soil simulations with various combinations of MAP, MAT, solar aspect, soil depth, soil hydraulic properties, and bedrock soil properties. The empirical relationship is appropriate for shallow bare soil overlying an open fracture in an impermeable bedrock.

The empirical relationship assumes that MAI can be parameterized as a function of the input variables (e.g., MAP, MAT, soil depth). A simple perturbation approach is used with a base set of the input variables to calculate a base value for MAI with a 1D simulation. Additional simulations were run, perturbing one or more input variables, to build up the response of MAI to the input variables. A simplified version of the empirical relationship derived by Stothoff et al. (1997) is used in UZFLOW for each pixel in a subarea:

Universal Transverse Mercator

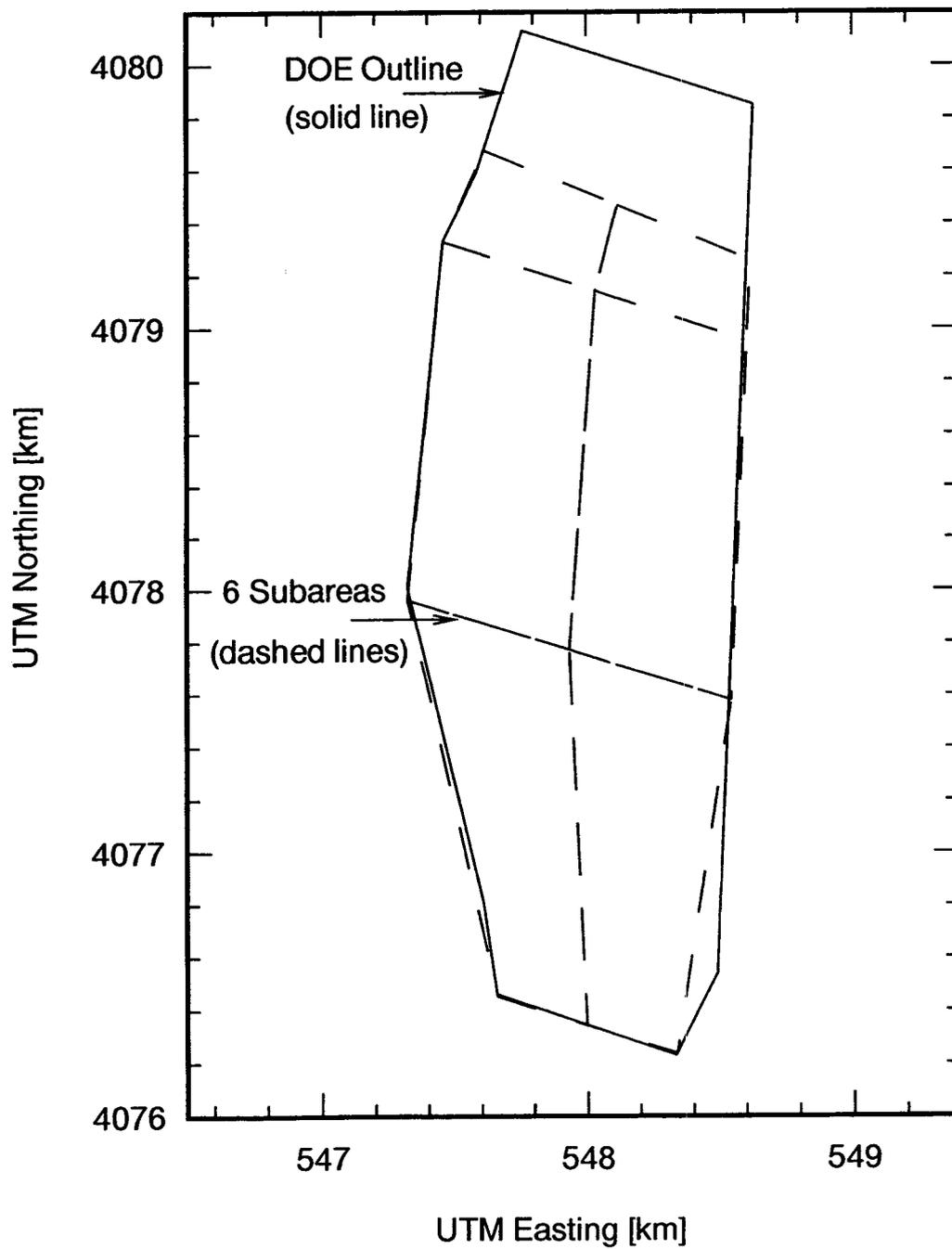


Figure 4-1. Example repository discretization into subareas

$$\begin{aligned} \log_{10} \left( \frac{(\text{MAI}/\text{MAP})}{0.77} \right) = & -0.005 + 0.307 \text{ MAV} \\ & + 1.63 \text{ P} - 12.6 \text{ T} - 1.73 \text{ P}^2 + 38.0 \text{ P T} + 141 \text{ T}^2 \\ & + 1.77 \text{ P}^3 - 44.2 \text{ P}^2 \text{ T} - 328 \text{ P T}^2 - 2710 \text{ T}^3 \\ & - 0.77 (15 \text{ b})^{-\sqrt{0.5}} \end{aligned} \quad (4-1)$$

where

MAI = mean annual infiltration [mm/yr]  
 MAP = mean annual precipitation [mm/yr]  
 MAT = mean annual temperature [C]  
 MAV = mean annual vapor density [gm/cm<sup>3</sup>]  
 P = ln(MAP/162.8)  
 T = [(MAT + 255.77)/290.53] - 1  
 b = soil depth [m]

The constants in the relationship account for representative YM soil and bedrock properties.

Under current climatic conditions, UZFLOW calculates MAP, MAT, and MAV for each pixel using regressed relationships (Stothoff et al., 1997):

$$\text{MAP} = \exp (4.26 + 0.000646 \text{ Z}) \quad (4-2)$$

$$\text{MAT} = 25.83 - 0.00840 \text{ Z} \quad (4-3)$$

$$\text{MAV} = \exp (-11.96 - 0.000341 \text{ Z}) \quad (4-4)$$

where

Z = ground surface elevation above sea level [m]

Ground surface elevation is supplied as a DEM in the static file entitled *ymelev.dem*; a DEM with the same discretization, *soildep.dem*, provides soil depths.

Climatic change, calculated by CLIMATO, is used to alter the value of MAP and MAT by changing the value of the parameter for each pixel by the same amount (e.g., if CLIMATO calculates that MAT is 1 C less than present, all pixels use a value of MAT that is 1 C cooler than calculated by Eq. 4-3). It is assumed that MAV remains unchanged from current conditions.

CLIMATO generates a sequence of MAP and MAT at uniform time steps. For each CLIMATO time step, UZFLOW uses the corresponding MAP and MAT to calculate MAI for each pixel within the subarea and averages all pixels to obtain an areal-average MAI, yielding a time history of MAI for each subarea. Cumulative MAI over time is interpolated to the time instants required by the user, yielding an average MAI between user time instants. Finally, each value of MAI in the time history is normalized to

match the average value of MAI over the repository footprint existing under current climatic conditions (input by the user).

## 4.2 NFENV

The near-field environment (NFENV) module calculates the time-dependent hydrothermal environment of the WP such as:

- Average repository-horizon rock temperature
- WP surface and spent fuel temperatures
- RH at the WP surface
- Flow rate of groundwater onto the WP
- pH and chloride concentrations of groundwater flowing onto the WP

The NFENV module requires as input the incoming water flow rate which is provided by the UZFLOW module. The temperatures, RH, dripping water flow rate, and chemical composition of the water are calculated and provided to the EBSFAIL and EBSREL modules.

The repository-horizon average rock temperature is computed using an analytic conduction-only model for mountain-scale heat transfer. The model is based on a heated rectangular region residing in a semi-infinite medium. The modeled repository region has been divided into 12 rectangular subregions to cover the proposed upper block of the repository. This discretization is shown in figure 4-2. The heated rectangular region is at a depth of H below the ground surface, and has width and length dimensions of 2B and 2L as shown in figure 4-3. Because more than one rectangular region exists, the temperature increase in the semi-infinite medium is the sum of contributions from each heated region. The general solution for the temperature increase at any point in space and time is given by (Claesson and Probert, 1996; Carslaw and Jaeger, 1959), and is:

$$\Delta T(x,y,z,t) = \int_0^t \frac{\alpha q''_{\text{rep}}(t')}{4k\sqrt{\pi}} \cdot \frac{1}{\sqrt{4\alpha(t-t')}} \left[ \operatorname{erf}\left(\frac{L-x}{\sqrt{4\alpha(t-t')}}\right) + \operatorname{erf}\left(\frac{L+x}{\sqrt{4\alpha(t-t')}}\right) \right] \cdot \left[ \operatorname{erf}\left(\frac{B-y}{\sqrt{4\alpha(t-t')}}\right) + \operatorname{erf}\left(\frac{B+y}{\sqrt{4\alpha(t-t')}}\right) \right] \cdot \left[ \exp\left(\frac{-z^2}{4\alpha(t-t')}\right) - \exp\left(\frac{-(z-2H)^2}{4\alpha(t-t')}\right) \right] \cdot dt' \quad (4.5)$$

where

$\Delta T(x,y,z,t)$  = increase in temperature at any time at any point in space and time in the semi-infinite medium due to one heated rectangular region [C]

$q''_{\text{rep}}(t)$  = time-dependent repository heat flux evaluated [W/m<sup>2</sup>]

$\alpha$  = thermal diffusivity of the semi-infinite medium [m<sup>2</sup>/s]

$k$  = thermal conductivity of the semi-infinite medium [W/(m-C)]

$L$  = half length of the heated rectangular region [m]

$B$  = half width of the heated rectangular region [m]

### Universal Transverse Mercator

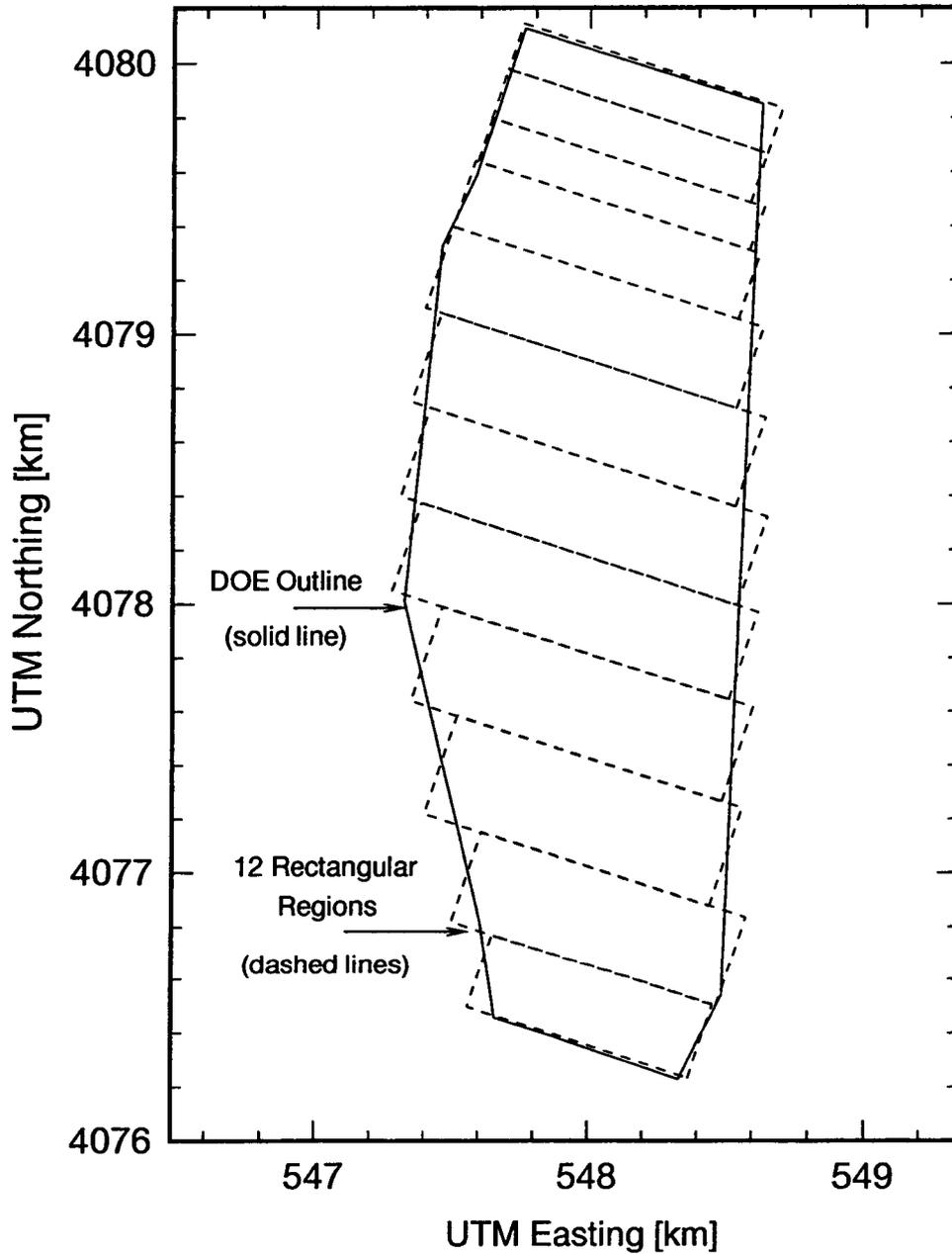


Figure 4-2. Discretization of repository upper block into twelve rectangular regions for the NFENV module

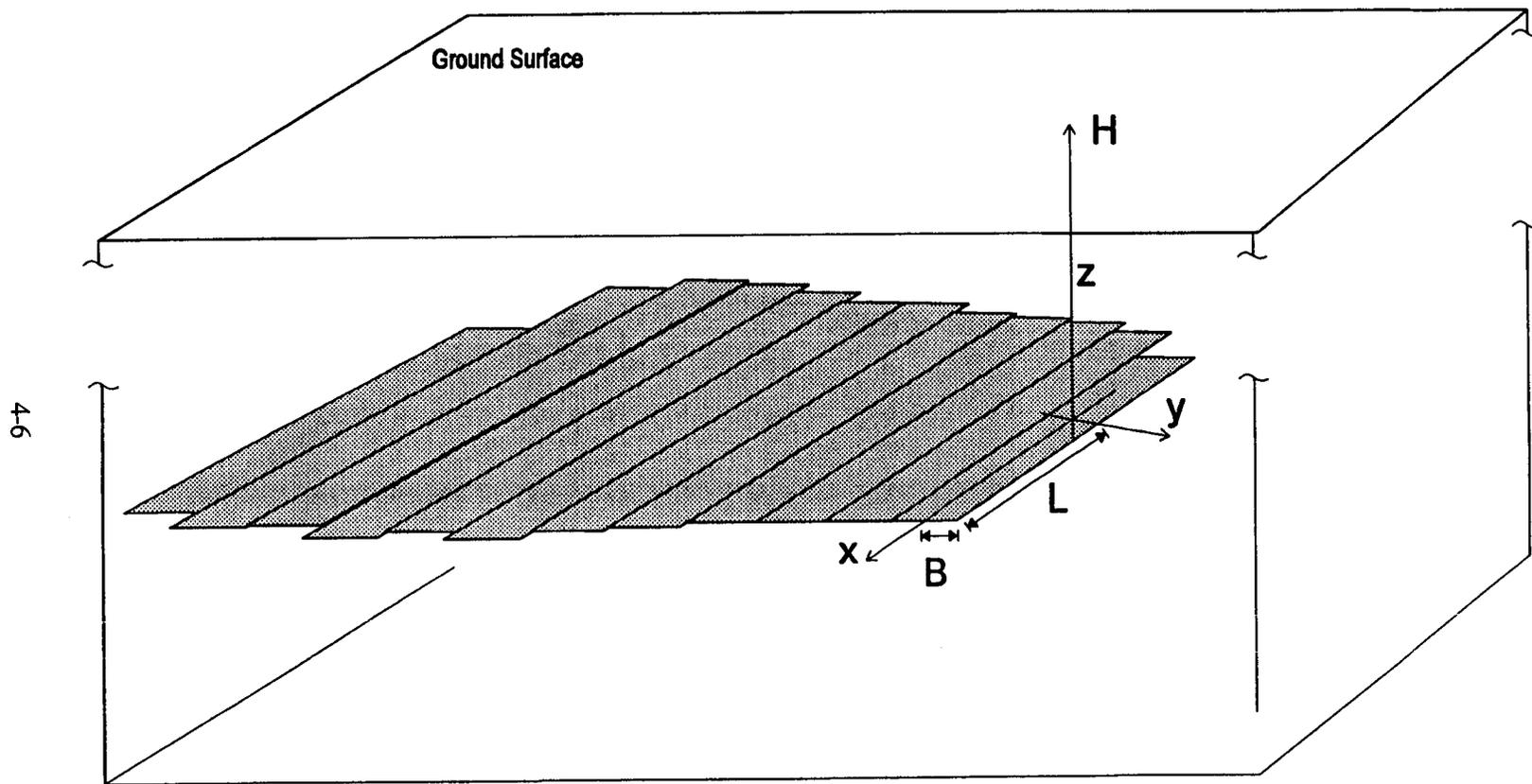


Figure 4-3. Mountain-scale heat transfer model with heated rectangular regions

H = depth of the heated region below the ground surface [m]  
 t = actual time after activation of heat flux [s]  
 t' = time of integration [s]  
 x,y,z = location of interest [m]

The ground surface is assumed to be exposed to atmospheric conditions and has a constant temperature (currently not affected by climate change). The analytic equation is valid below the ground surface,  $z < H$ . The repository heat flux is related to the AML and heat output per MTU of waste:

$$q''_{\text{rep}}(t) = \text{AML} \cdot Q_{\text{per mtu}}(t) \quad (4-6)$$

Likewise, the thermal output for a single WP is related to the WP payload:

$$Q_{\text{wp}}(t) = \text{MTU}_{\text{wp}} \cdot Q_{\text{per mtu}}(t) \quad (4-7)$$

where

AML = areal mass loading [MTU/m<sup>2</sup>]  
 MTU<sub>wp</sub> = metric tons of uranium [MTU] in a representative WP  
 Q<sub>per mtu</sub>(t) = time-dependent heat output per MTU of waste [W/MTU]

The time-dependent heat output of the waste per MTU [Q<sub>per mtu</sub>(t)] is available using a function subroutine in the INVENT utility module.

The repository upper block has been discretized into 12 rectangular subregions to cover the extent of the proposed upper block, as shown in figure 4-2. Waste is assumed to be emplaced uniformly throughout the rectangular subregions, so there is no spatial variation in the waste heat output. This assumption should be re-evaluated in the future to explore the effects of spatially varying thermal loads. The temperature increase at any point is due to the contribution from all subregions. The average rock temperature is computed at an elevation of half the drift diameter at the center of each of the subareas. The subareas should not be confused with the NFENV rectangular regions. The rectangular regions are used by the NFENV module to predict only the temperatures in the subareas. The analytic mountain-scale conduction model predicts the rock-wall temperature ( $T_{\text{rock}}$ ) as a function of time. Having computed  $T_{\text{rock}}$  for the subarea, the WP temperature can be calculated.

A multimode (i.e., conduction, convection, and radiation) heat transfer model is used for modeling drift-scale heat transfer. A simplified thermal network is shown in figure 4-4 for heat transfer from the waste to the drift rock wall. The network consists of four temperature nodes and five thermal conductance linkages (conductance is the reciprocal of resistance). The thermal network is used to predict the WP surface temperature and the maximum spent fuel temperature given  $T_{\text{rock}}$  (from the mountain-scale model described above) and  $Q_{\text{wp}}(t)$ .

The method used to solve the thermal network problem is to progressively solve each of the three unknown temperatures. The first is the WP surface temperature ( $T_{\text{wp,surf}}$ ), then the inner wall surface temperature ( $T_{\text{in,surf}}$ ), and finally the maximum spent fuel temperature ( $T_{\text{max,sf}}$ ). The heat is transferred by both thermal radiation and natural convection in the unbackfilled region above a WP and by conduction

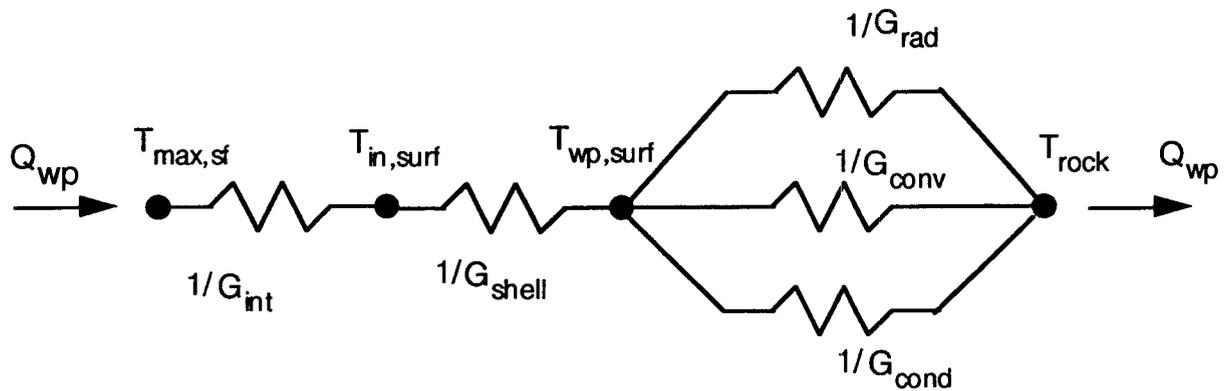


Figure 4-4. Thermal network used to predict the maximum spent fuel temperature

through the package support and floor material, and can be expressed as:

$$Q_{wp} = (G_{rad} + G_{conv} + G_{cond}) (T_{wp,surf} - T_{rock}) \quad (4-8)$$

where

$T_{wp,surf}$	= WP surface temperature [C]
$T_{rock}$	= near-field rock temperature [C]
$Q_{wp}$	= time dependent heat output for a WP [W]
$G_{rad}$	= effective thermal conductance for radiative heat transfer [W/C]
$G_{conv}$	= effective thermal conductance for convective heat transfer [W/C]
$G_{cond}$	= effective thermal conductance for conduction [W/C]

Equation (4-7) is used to solve for WP surface temperature, given the rock temperature, thermal output of the WP and conductances (Gs). The thermal conductance for radiative transfer above the WP is based on a linearization of the Stefan-Boltzmann law, and accounts for the emissivity of the WP and drift rock wall (Incropera and DeWitt, 1990):

$$G_{\text{rad}} = \left( \frac{3}{4} \right) \frac{4\sigma(T_{\text{rock}})^3}{\frac{1 - \epsilon_{\text{wp}}}{\epsilon_{\text{wp}} \pi D_{\text{wp}} L_{\text{wp}}} + \frac{1}{F_{\text{wp-rw}} \pi D_{\text{wp}} L_{\text{wp}}} + \frac{1 - \epsilon_{\text{rw}}}{\epsilon_{\text{rw}} \pi D_{\text{rw}} L_{\text{rw}}}} \quad (4-9)$$

where

- $\sigma$  = Stefan-Boltzmann constant [=5.67 10<sup>-8</sup> W/(m<sup>2</sup>-K<sup>4</sup>)]
- $\epsilon_{\text{wp}}$  = emissivity of the WP surface [unitless]
- $\epsilon_{\text{rw}}$  = emissivity of the drift rock wall surface [unitless]
- $F_{\text{wp-rw}}$  = radiative view factor from the WP to the rock wall (=1) [unitless]
- $D_{\text{wp}}$  = diameter of the WP [m]
- $D_{\text{rw}}$  = diameter of the rock wall [m]
- $L_{\text{wp}}$  = length of the WP [m]
- $L_{\text{rw}}$  = length of drift wall per WP drift [m] (estimated to be ~18 m for 80 MTU/acre, ~25 m for 40 MTU/acre, and ~30 m for 25 MTU/acre)

The top three-quarters of the WP is available for the radiative/convective heat transfer and the bottom quarter of the package participates in conduction through the pedestal/floor. The thermal conductances for convective transfer above the WP and conductive transfer below the package are computed from:

$$G_{\text{conv}} = \left( \frac{3}{4} \right) \frac{2\pi k_{\text{eff,air}}(2L_{\text{wp}})}{\ln(D_{\text{rw}}/D_{\text{wp}})} \quad (4-10)$$

$$G_{\text{cond}} = \left( \frac{1}{4} \right) \frac{2\pi k_{\text{floor}}(2L_{\text{wp}})}{\ln(D_{\text{rw}}/D_{\text{wp}})} \quad (4-11)$$

where

- $k_{\text{eff,air}}$  = effective thermal conductivity of buoyant air which has been estimated to be 30 times the stagnant air conductivity [W/(m-C)] (Manteufel, 1997)
- $k_{\text{floor}}$  = thermal conductivity of the concrete pedestal/floor material [W/(m-C)]

The effective axial length for conductive and convective transfer from the WP to the drift wall should be larger than the length of the WP and smaller than the package spacing length within a drift. A reasonable value for this length is two times the WP length, and this value is used in the preceding equations. For the case of backfilled drifts, heat transfer through the top three quarters is predicted using an effective conductivity for the crushed, backfill material.

$$G_{bf} = \left( \frac{3}{4} \right) \frac{2\pi k_{eff,bf}(2L_{wp})}{\ln(D_{rw}/D_{wp})} \quad (4-12)$$

where

- $G_{bf}$  = effective thermal conductance for backfill region [W/C]  
 $k_{eff,bf}$  = effective thermal conductivity of backfill material [W/(m-C)]

After computing the outer WP surface temperature, the inner surface temperature of the wall is calculated. The wall of the WP consists of two cylindrical layers for the inner and outer overpacks. The thicknesses and properties of the walls are specified as sampled parameters in the input file. The inner wall temperature is related to the WP heat according to:

$$Q_{wp} = G_{shell} (T_{in,surf} - T_{wp,surf}) \quad (4-13)$$

where

- $G_{shell}$  = thermal conductance for WP shell [W/C]  
 $T_{in,surf}$  = inner surface temperature of the package wall [C]

The shell conductance consists of a contribution from the outer carbon steel and inner stainless steel layers.

$$G_{shell} = \frac{L_{wp}}{\frac{t_{ss}}{\pi D_{ss} k_{ss}} + \frac{t_{cs}}{\pi D_{cs} k_{cs}}} \quad (4-14)$$

where

- $t_{ss}$  = thickness of the inner stainless steel layer [m]  
 $t_{cs}$  = thickness of the outer carbon steel layer [m]  
 $D_{ss}$  = diameter of the inner stainless steel layer [m]  
 $D_{cs}$  = diameter of the outer carbon steel layer [m]  
 $k_{ss}$  = thermal conductivity of stainless steel [W/(m-C)]  
 $k_{cs}$  = thermal conductivity of carbon steel [W/(m-C)]

Using the inner surface temperature, the maximum spent fuel temperature is calculated using a conduction shape factor formula which accounts for the volumetric heat generation in the interior region of the package (Manteufel and Todreas, 1994).

$$Q_{wp} = G_{int} (T_{max,sf} - T_{in,surf}) \quad (4-15)$$

where the conductance of the cylindrical interior region is computed from:

$$G_{\text{int}} = k_{\text{sf}} S L_{\text{wp}} \quad (4-16)$$

where

- S = conduction shape factor for a heated cylindrical region [=  $4\pi$ ]  
 $k_{\text{sf}}$  = effective thermal conductivity of basket and spent fuel in the package [W/(m-C)]

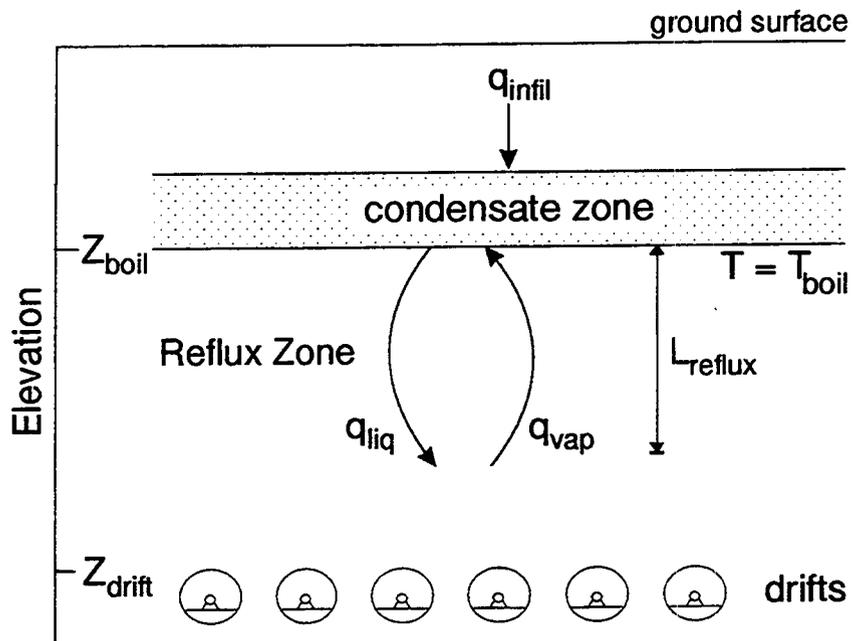
The effective thermal conductivity of the spent fuel accounts for the region between the inner wall and the basket material, the basket material, and the individual assemblies. There are multiple modes of heat transfer, including thermal radiation, buoyant convection primarily in the larger void regions, and conduction in the basket material, the fuel rods, and regions with primarily stagnant gas. At high temperatures, the heat transfer is dominated by radiative transfer. At lower temperatures, heat transfers can be dominated by conduction. The effective thermal conductivity is a function of temperature and increases with higher temperatures. In this work, the effective thermal conductivity is a sampled parameter and is specified in the *tpa.inp* file. In future work, a temperature dependent model should be included in NFENV.

In addition to the thermal calculations, NFENV provides estimates of groundwater infiltration rates and chemical composition (i.e., pH and chloride concentration) for the flow onto WPs. The pH and chloride concentrations are based on table-lookup using results from the MULTIFLO code (Lichtner and Seth, 1996). The groundwater flow rates onto a WP are internally calculated in the NFENV module. MULTIFLO results are multiplied by a factor which is a sampled parameter to represent uncertainties in the data.

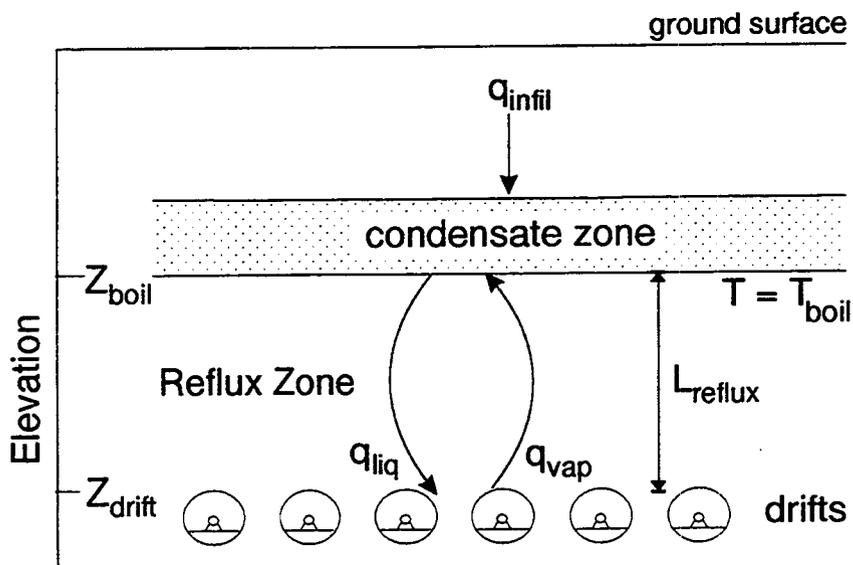
NFENV uses the time-dependent temperature profiles generated by these heat transfer models, along with time-dependent water flux ( $q_{\text{infil}}$ ), to calculate time-dependent water flux ( $q_{\text{drip}}$ ) dripping onto a WP. In the development of  $q_{\text{drip}}$ , NFENV considers (i) the time-dependent amount of perching due to thermal pulsing, (ii) time-dependent refluxing of liquid and vapor, and (iii) drift-scale variability of hydraulic properties and fluxes.

The thermohydrologic conceptual model implemented in NFENV assumes that there are both matrix and fracture flow continua. It is assumed that a condensate zone layer exists at a temperature above the boiling point  $T_{\text{boil}}$  isotherm with a thickness dependent upon  $q_{\text{infil}}$ , as shown in figure 4-5. Below the  $T_{\text{boil}}$  isotherm is a reflux zone with thickness  $L_{\text{reflux}}$ . Above the  $T_{\text{boil}}$  isotherm, liquid is supplied to the fractures at a rate proportional to the thickness of the condensate zone layer. In the reflux zone, liquid from the condensate zone flows down through fractures and is vaporized (since  $T > T_{\text{boil}}$ ). The vapor rises to the top of the boiling zone and condenses back to liquid in the condensate zone. The thickness of the reflux zone is dependent on  $q_{\text{infil}}$  and the local heat flux, that is, the temperature gradient. When the value of  $L_{\text{reflux}}$  subtracted from the elevation of the  $T_{\text{boil}}$  isotherm,  $Z_{\text{boil}}$ , is below the elevation of the top of the drift, water begins to drip into the drift. Any liquid passing below the level of the repository is assumed to continue to the water table, and the thickness of the perched zone is decreased accordingly.

The near-field thermal response to the heat pulse is assumed to be dominated by conduction heat transfer and the near-field hydrology response is dominated by the temperature distribution. It is also assumed that the near-field moisture distribution reaches equilibrium rapidly relative to changes in the temperature field. Three input variables are required: (i)  $q_{\text{infil}}$ , the infiltration flux into the subarea, (ii)  $Z_{\text{boil}}$ , the elevation of the boiling isotherm above the repository, and (iii)  $q_{\text{heat}}$ , the thermal flux at  $Z_{\text{boil}}$ .



(a) no dripping



(b) dripping

Figure 4-5. Conceptualization of drift-scale thermal hydrologic model

A mass balance model is used to model the thickness of the perched water zone:

$$(\theta - \theta_r) \frac{\partial L_p}{\partial t} = q_{\text{infil}} - q_{\text{perc}} \quad (4-17)$$

where

- $\theta$  = moisture content [unitless]
- $\theta_r$  = residual moisture content [unitless]
- $L_p$  = thickness of the perched zone [m]
- $q_{\text{perc}}$  = percolation flux at the repository level [m/yr]
- $t$  = time [yr]

After it is determined that dripping in a drift can occur (i.e.,  $Z_{\text{boil}} - L_{\text{reflux}} < Z_{\text{drift}}$ ), the dripping flux is calculated. Since flow in the unsaturated zone is considered to be primarily through fractures but not all fractures have flow,  $f_{\text{local}}$  is defined as the fraction of a subarea having fracture flow. As WPs cannot be dripped on if there is no flow, the fraction of drifts in which dripping occurs is also represented by  $f_{\text{local}}$ . The estimated value of volumetric flux of water into a drift given that dripping occurs ( $E[q_{\text{drip}} | q_{\text{drip}} > 0]$ ) is therefore represented as:

$$E[q_{\text{drip}} | q_{\text{drip}} > 0] = \frac{q_{\text{perc}}}{f_{\text{local}}} \quad (4-18)$$

where  $q_{\text{perc}}$  is the percolation volumetric flux through the unsaturated zone above the drift as determined by the UZFLOW module.

In drifts with  $q_{\text{drip}} > 0$  water can (i) drip on a WP or (ii) drip elsewhere, missing emplaced WPs. The fraction of  $q_{\text{drip}}$  that drips on a WP is represented by  $f_{\text{hit}}$ , a sampled parameter that represents the fraction of the dripping flux which will enter the WP after it fails. EBSREL further partitions the flow contacting WPs into the portions contacting failed WPs and contacting WPs that have not failed, while EBSFAIL determines the fraction of WP that has failed due to corrosion. For release of contaminants from WP modeled by EBSREL, the volume of water available for dissolving radionuclides is then:

$$q_{\text{drip}} = E[q_{\text{drip}} | q_{\text{drip}} > 0] \times f_{\text{hit}} \quad (4-19)$$

In summary, NFENV provides temperature, RH, groundwater flow rate, and chemical properties information for determination of corrosion rates and WP failures. It also provides the flow rate of water onto a WP to the EBSREL module for dissolution of the WP contents following WP failure.

### 4.3 EBSFAIL

The engineered barrier system failure (EBSFAIL) module calculates the failure time of the EBS due to various modes of degradation similar to the SOTEC code (Sagar et al., 1992) used in IPA Phase 2. EBSFAIL executes the stand-alone FAILT program which is a part of the engineered barrier system performance assessment codes (EBSPAC) by Mohanty et al. (1996). These modes of degradation include dry oxidation, uniform corrosion, localized (pitting and crevice) corrosion, and fracture failure. Other degradation modes that may become important under certain conditions, such as stress corrosion cracking and microbially influenced corrosion, are not currently considered in EBSFAIL. Inputs required by EBSFAIL include the chemical composition of the fluid in contact with a WP and information on the temperature and RH as a function of time and position in the EBS (all of this information is calculated by NFENV). The output of EBSFAIL is the WP corrosion failure time history which is provided to EBSREL.

Three different types of WP failure are considered in EBSFAIL (i) initial failure, (ii) disruptive scenario failure, and (iii) corrosion and mechanical failures. These failures are referred to as Type 1, Type 2, and Type 3 failures, respectively. In Type 1 failure, a portion of the WPs in a cell is specified to have failed at time  $t = 0$  yr as a result of initial defects produced before repository closure. These WPs are assumed to have been defective or damaged prior to or during emplacement and are specified in the input data as a fraction of the total containers in a cell. In Type 2 failure, WPs fail as a result of some disruptive event. The timing and number of WPs affected by Type 2 failure are calculated by other consequence modules (e.g., FAULTO, SEISMO, VOLCANO).

All WPs in a subarea that have not undergone Type 1 and 2 failures are potentially subjected to corrosion and mechanical (Type 3) failures. This assumption implies that corrosion or mechanical failure affects all WPs equally in a cell so that when one WP fails, then all WPs in the same cell that have not already failed under Type 1 and 2 modes will fail simultaneously. For simplicity, failure of the WP is defined as the through-wall penetration of the outer and inner overpacks by a single pit or by uniform corrosion. Failure can also occur by brittle fracture due to mechanically dominated processes. No allowance is given to the protection ability of the multipurpose container (MPC) or the fuel cladding against corrosion or mechanical failure. After the outer and inner overpacks are penetrated or failed by fracture, the spent fuel (SF) is considered to be completely exposed to the near-field environment.

Two types of near-field environments are considered, leading to different corrosion processes. One is hot dry air that promotes oxidation of the outer steel overpack, and the other is humid air at a high RH that induces aqueous corrosion. The presence of  $\text{Cl}^-$  in the environment may promote localized corrosion under slightly alkaline conditions ( $\text{pH} > 8.0$ ).

Water condensation begins when the temperature at the WP surface decreases to a value at which the RH of the environment surrounding the WP reaches a threshold or critical RH. The thickness of the condensed liquid layer is assumed to be the same regardless of the presence or absence of backfill material around the WP and the nature of the contact between particles of backfill and the WP surface. It is assumed that the wettability of this surface is such that water droplets impinging on it or condensing nuclei of water vapor can spread immediately to form a layer of uniform thickness. The water layer is thin and defined by an arbitrary, specified thickness on the order of a few mm.

At every time step, a calculation is performed to determine if the RH has reached the critical value. The environment surrounding the WP is treated as dry air if the RH is lower than the threshold, and an amount of outer overpack material consumed by dry oxidation is calculated to determine the penetration of the oxidation front. In the same time interval, a mechanical failure test is conducted for the new thickness resulting from metal oxidation to evaluate if failure due to mechanical fracture occurs. If the WP does not fail by fracture, then the time is advanced, and the same test is repeated. If at any time step water condensation takes place, then the calculation of oxidation in dry air is interrupted, and the aqueous corrosion calculation is initiated. The mechanical failure test is performed at all time intervals until failure occurs, regardless of whether the WP is undergoing dry oxidation or aqueous corrosion.

Aqueous corrosion could be uniform or localized. If it is localized because the corrosion potential is above the critical potential for the initiation of localized corrosion, the calculation of penetration is initiated immediately in the form of pit growth without assuming an initiation or induction time. When the depth of the pit is greater than the initial thickness of the WP outer overpack, the potential of the galvanic couple formed by the outer and inner container is calculated. If the corrosion potential of the couple or galvanic potential is lower than the critical potential for localized corrosion of the inner container, penetration of the inner container is computed as uniform corrosion under passive conditions. Otherwise, pit growth of the inner overpack begins and continues until the depth of the pit becomes equal to the inner overpack wall thickness.

### 4.3.1 Dry Air Oxidation

Oxidation of steel can take place under relatively dry conditions in the presence of air at relative humidities lower than 70 percent and temperatures ranging from ambient up to 250 °C. The oxide layers formed at such temperatures are considered to protect the container against further oxidation. However, the possibility exists that oxide growth may become localized. Localized dry air oxidation may lead to a deeper oxidation that may adversely affect the long-term container integrity in a dry air environment.

Localized dry oxidation includes internal oxidation and intergranular oxidation. In the case of internal oxidation, the oxide forms as islands in the metal underneath the uniform oxide layer. In intergranular oxidation, the oxide forms preferentially along grain boundaries. Localized dry oxidation takes place by mass transport through short-circuit diffusion paths, such as interfaces between metal and oxide (or other inclusions and precipitates) or grain boundaries. Therefore, localized dry oxidation can penetrate into the metal deeper than uniform dry oxidation.

For the calculations of intergranular oxide formation, a mathematical model developed by Oishi and Ichimura (1979) is used, in which oxygen diffusion in the matrix and along the grain boundary in an infinite one-dimensional (1D) body is calculated simultaneously. The main assumptions in the calculations are (i) negligible effects of external oxide, (ii) diffusion of oxygen into metallic phases (near the interface between grain boundary oxide and metal), and (iii) diffusion of oxygen into metallic matrices (from grain boundaries). The distance of oxygen penetration in the metal is then represented by:

$$Y_p = \left[ \frac{4 D_1}{r_g \delta D_g} \sum_{n=1}^{\infty} \exp \left( - \frac{D_1 n^2 \pi^2 t}{r_g^2} \right) \right]^{-1/2} \quad (4-20)$$

where

- $Y_p$  = penetration distance by intergranular oxidation [cm]
- $D_1$  = matrix diffusivity [ $\text{cm}^2/\text{s}$ ]
- $D_g$  = grain boundary diffusivity [ $\text{cm}^2/\text{s}$ ]
- $\delta$  = the thickness of grain boundary [ $\approx 0.7 \times 10^{-7}$  cm, based on Lobnig et al., 1992]
- $r_g$  = the grain radius [ $\approx 1 \times 10^{-3}$  cm for cast steel, based on Ahn and Soo, 1983; 1984]
- $t$  = time [s]

Equation (4-19) yields penetration distance of oxygen along grain boundaries and is being used as a surrogate for oxide formation.

### 4.3.2 Aqueous Corrosion

Aqueous corrosion takes place only when the metal surface is covered by a water film. Water can be physically adsorbed to the metal surface in molecular form or it can be chemically bonded in a dissociated form, which results in the formation of metal-hydroxyl bonds (Leygraf, 1995). As shown in table 4-1, RH determines the characteristics and thickness of the water film. The critical RH above which atmospheric corrosion of most metals occurs, closely coincides with the RH necessary for the formation of multiple water monolayers and the liquid film behaves in a manner similar to bulk water. Under these conditions, corrosion is governed by the same electrochemical laws applicable to corrosion of metals immersed in an aqueous electrolyte.

Several factors can decrease the critical RH required to form a multilayer surface water film. Particulate matter from the air can deposit on the surface and promote the adsorption of water (a similar effect can be expected from particles of backfill material). Similarly, the deposition of hygroscopic salts on the metal surface can substantially decrease the critical RH necessary to form a water film or capillary condensation of water can occur in the pores of a thick oxide layer. These lower order processes are not considered in TPA Version 3.0 code.

Iron and steel exhibit a primary critical RH of around 60 percent, similar to most metals (Fyfe, 1994). Above 60 percent RH, corrosion proceeds at a slow rate, but at 75–80 percent RH the corrosion rate sharply increases. This secondary critical RH is attributed to capillary condensation of water in the pores of the solid corrosion products. The water films that form on the metal surface usually contain a variety of contaminants, including trace amounts of  $\text{Cl}^-$  and other soluble species such as  $\text{CO}_2$  that increase the electrical conductivity and decrease the pH of the film, and lead to an increase in the dissolution of the iron or steel (Leygraf, 1995).

Below a critical value of RH, air oxidation of steel is modeled as the dominant corrosion process for the steel overpack. If the RH is higher than the critical value, the occurrence of aqueous corrosion of

**Table 4-1. Approximate number of water monolayers versus relative humidity (Leygraf, 1995)**

RH (percent)	Number of Water Layers
20	1
40	1.5-2
60	2-5
80	5-10

the steel overpack is evaluated. No distinction is made between humid air corrosion and aqueous corrosion because both processes are governed by the same electrochemical kinetics mechanisms.

In the presence of an aqueous phase, corrosion of steel is an electrochemically controlled process. This process could be relatively uniform (as active dissolution at pH lower than neutral) or localized (under slightly alkaline conditions) promoting passivity in the presence of aggressive anions such as  $\text{Cl}^-$ . The corrosion process at any given time depends on the corrosion potential and the critical potential required to initiate a particular localized corrosion process. In this analysis, the repassivation potential,  $E_{rp}$ , is conservatively adopted as the critical potential for the initiation of localized corrosion. The same approach is applied to the outer and inner containers assuming that steel behaves as a corrosion-resistant alloy in alkaline, passivating environments. If the corrosion potential is higher than the repassivation potential, it is assumed that localized corrosion is initiated without an induction time; if not, uniform corrosion under passive conditions takes place. The corrosion models calculate the penetration or remaining thickness at each time step using rates of uniform and localized corrosion.

The corrosion models calculate the rates of uniform wet corrosion and localized corrosion (pitting corrosion) following the approach adopted previously in the SCCEX code (Cragnolino et al., 1994). The dominant corrosion process at any given time is dictated by the corrosion potential and the appropriate critical potential for that process. The corrosion potential is the mixed potential established at the metal/solution interface when a metal is immersed in a given environment. Corrosion potentials are calculated on the basis of kinetic expression for the cathodic reductions of oxygen and water and the passive current density for the anodic oxidation of the metals. If the corrosion potential exceeds the critical potential for pit initiation, pits are assumed to initiate and grow without an initiation time. If the corrosion potential falls below the repassivation potential, previously growing pits are assumed to cease growing and the material passivates, corroding uniformly at a very low rate through a passive film.

Empirically derived equations are used in EBSFAIL for the dependence of critical potentials on environmental parameters. The pit initiation and repassivation potentials are assumed to depend only on the chloride concentration and temperature. The dependence of the critical potential on chloride concentration and temperature is given by:

$$E_{\text{crit}} = E_{\text{crit}}^{\circ}(T) + B(T) \log [\text{Cl}^{-}] \quad (4-21)$$

where the quantities  $E_{\text{crit}}^{\circ}(T)$  and  $B(T)$ , are linear functions of temperature and dependent on the material. It should be noted that  $E_{\text{crit}}^{\circ}(T)$  is the value of  $E_{\text{crit}}(T)$  for a  $\text{Cl}^{-}$  concentration equal to 1 M. Both  $E_{\text{crit}}^{\circ}(T)$  and  $B(T)$  were evaluated for A516 steel and Alloy 825 from literature and CNWRA data for initiation and repassivation potentials for both pitting and crevice corrosion (Sridhar et al., 1993; Dunn et al., 1996).

Following penetration of the outer container, electrical contact of the inner and outer container through the presence of an electrolyte path (such as that provided by modified groundwater) promotes galvanic coupling, assuming that metallic contact always exists between both containers. The galvanic coupling model evaluates whether penetration of the inner container by localized corrosion is possible; if not, uniform corrosion or mechanical fracture becomes the predominant failure mechanism because the inner container becomes protected against localized corrosion.

The effect of galvanic coupling between the inner and the outer overpacks on the failure time of the WP is evaluated by a simplified approach. The corrosion potential of the galvanic couple formed when the wall of the outer container is penetrated by a pit,  $E_{\text{corr}}^{\text{WP}}$ , is estimated using experimentally measured values of the potential bimetallic couple,  $E_{\text{couple}}$ , for a well-defined area ratio between both components. If  $E_{\text{corr}}^{\text{WP}}$  is greater than the repassivation potential of the inner overpack material, localized corrosion occurs. Otherwise, uniform corrosion takes place. The  $E_{\text{corr}}^{\text{WP}}$  is determined through a linear combination of  $E_{\text{corr}}$  of the inner overpack, in the absence of galvanic coupling at the time of the through-wall penetration of the outer overpack, and  $E_{\text{couple}}$  according to the following assumed empirical expression:

$$E_{\text{corr}}^{\text{WP}} = (1-\eta) E_{\text{corr}} + \eta E_{\text{couple}} \quad (4-22)$$

where  $\eta$  is the efficiency of the galvanic coupling with the condition  $0 \leq \eta \leq 1$ . A value of  $E_{\text{couple}}$  equal to  $-0.46V_{\text{SHE}}$  (volts referenced to standard hydrogen electrode scale) was adopted on the basis of results reported by Scully and Hack (1984) for a galvanic couple made of steel and alloy 625 (a nickel-base alloy similar in electrochemical behavior to Alloy 825) with an area ratio 1:1 and exposed to sea water. The values adopted for the different parameters needed to calculate  $E_{\text{corr}}$  and those establishing the dependence of the critical potentials with chloride concentration and temperature are reported elsewhere (Mohanty et al., 1996).

### 4.3.3 Fracture Failure

The possibility of mechanical failure as a result of thermal embrittlement of the steel promoted by long-term exposure to temperatures above 150 °C is evaluated at each time step. Fracture, as a result of thermal embrittlement of the steel overpack, is an important failure mode to be considered for any WP

design, particularly for high thermal loadings. Thermal embrittlement of low-alloy steels occurs as a consequence of prolonged exposure at elevated temperatures and results in a substantial degradation of specific mechanical properties.

One of the important mechanical properties required for a WP material is toughness, which is the ability to absorb energy in the form of plastic deformation without fracture. However, toughness is significantly affected by thermal embrittlement, a phenomenon closely related to temper embrittlement. This type of embrittlement is characterized by an upward shift in the ductile-brittle transition temperature, measured by the variation of the impact fracture energy for notch specimens as a function of test temperature (Vander Voort, 1990). Segregation of impurities, such as Sb, P, Sn, and As, along prior austenite grain boundaries is the main cause of temper embrittlement (Briant and Banerji, 1983). The segregation of P, which in the case of commercial steels is the predominant impurity, promotes fracture of notched specimens upon impact and leads to a change in the low-temperature fracture mode from transgranular cleavage to intergranular fracture.

Mechanical failure of WP in the EBSFAIL module is considered to be the result of fracture of the outer steel overpack. As a first approximation, other mechanical failure processes such as buckling or yielding are not considered plausible for the current design of the WP due to the relatively large thickness of the container wall. It should be noted that active uniform corrosion of the carbon steel overpack is not expected under the passivating conditions prevailing in the near-field environment and therefore, failure modes such as buckling or yielding that would require significant generalized thinning of the container wall in the presence of external loads were not included in the analysis.

A simple fracture model, based on a generalized expression for the stress intensity factor ( $K_I$ ) developed on the basis of linear-elastic fracture mechanics, is used in EBSFAIL. For the case of a cylinder with a surface flaw located on its outer surface, the following equation is applicable:

$$K_I = Y \sigma (\pi a)^{0.5} \quad (4-23)$$

where

- $K_I$  = stress intensity factor for the crack opening mode (I), [MPa m<sup>0.5</sup>]
- $Y$  = geometry factor to account for the shape of the crack and the load configuration [unitless]
- $\sigma$  = applied stress [MPa]
- $a$  = depth of crack [m]

It is assumed that applied stresses are due only to residual stresses associated with the circumferential weld used for overpack closure. Following a commonly accepted criterion, it is assumed that the maximum value attainable by residual stresses produced by welding is the yield strength of the material (currently set in the FAILT stand-alone code to 205 MPa). The depth of the crack increases with time as a result of localized corrosion in the form of a pit, which is conservatively assumed to be equivalent to a crack. Values of  $Y$  are calculated as presented by Rolfe and Barson (1977). The factor  $Y$  corresponds to a part-through thickness thumbnail crack with a length  $2c$  equal to two times its depth  $a$  for a hollow cylinder of wall thickness  $t$  in which the crack shape parameter  $Q$  is a function of  $\frac{a}{2c}$ . A

magnification factor  $M_K$  varying from 1.0 to 1.6 was introduced for deep cracks with depths ranging from  $t/2$  to  $t$ . The Y factor is defined using the nomenclature of Rolfe and Barson (1977) as  $Y = M_K Q^{-0.5}$ . For simplicity, the WP is considered to be composed of a single shell with the added thickness of both the outer and the inner overpack but with the mechanical properties of the outer overpack.

In addition, a safety factor of 1.4 was applied to calculate the value of  $K_I$  by assuming that the yield strength of the material in the vicinity of the welds is higher than the base material. This value is compared at each time step with the critical stress intensity or fracture toughness of the material,  $K_{Ic}$ , to determine if failure by fracture takes place. By definition, fracture occurs instantaneously if  $K_I$  is greater than  $K_{Ic}$ . Due to the lack of data, no decrease in the value of  $K_{Ic}$  with time is assumed in the present analysis, which may be the case if thermal embrittlement of the steel occurs due to prolonged exposure (thousands of years) to temperatures above 250 °C.

#### 4.4 EBSREL

The engineered barrier system release (EBSREL) module calculates the time-dependent release of radionuclides after the EBSFAIL module determines the WP has been breached, using temperature, chemical composition of the fluid, and liquid flow rate information provided by NFENV. EBSREL executes the RELEASET stand-alone program which is part of EBSPAC (Mohanty et al., 1996). EBSREL takes into account radionuclide decay, generation of daughter products in the chains, temporal variation of inventory in the WP, and spatial variations in the properties of the surrounding material. Two conditions must be satisfied for a release: (i) a WP must fail and (ii) a liquid environment must be available around the WP at temperatures below the boiling point of water. Time-dependent release rates are calculated for each radionuclide and this is provided to UZFT. EBSREL considers only radionuclide releases from SF. Since the WPs are assumed to contain only SF, no consideration of radionuclide release from glass waste form is made in this module.

The first step in the calculation of liquid releases is to determine if a liquid release is possible at a given time. The release calculation at every time step includes the computation of the radionuclide inventory in the solid mass, radionuclide releases from the solid mass into the liquid surrounding the WP, the generation of the new radionuclide inventory in the liquid due to radioactive ingrowth, convective release of mass from inside to the outside of the WP, and diffusive losses into the surrounding medium outside the WP.

At each time step, the inventory of the radionuclides in the water inside the WP is computed, and the radioelement inventory is computed as the sum of mass of all the isotopes. If the concentration of that radioelement in the WP water exceeds its solubility limit, then the calculated concentration value is discarded, and the solubility limit is assigned to the concentration at that time step. At any given time, the concentration of a nuclide in the WP water is calculated by dividing the element inventory by the volume of water in the WP. The release of an individual nuclide occurs at a rate that is proportional to the mass fraction of all the isotopes of the radioelement.

At the end of this calculation, the cumulative release is recorded for each nuclide. After advancing the time, the calculation is repeated for the next time step. The calculation continues until all radionuclides are depleted from the solid SF and the water in the WP or the end of the simulation is

reached, whichever comes first. Since release calculations focus on release from a single WP, to obtain the total release in the final calculations, the release from one WP is multiplied by the total number of wetted WPs for each failure type in a subarea.

It is assumed that at the failure time there are at least two pits acting as conduits in a horizontally emplaced WP, which are located such that water enters through one pit and exits through the other one. Another assumption is that one of the pits is located on the side of the WP at a level lower than that of the water entrance pit, which is situated at the top of the horizontally emplaced WP. After the water level in the WP rises to the specified outflow position (a sampled parameter), water begins to flow from the WP along with the dissolved radionuclides.

Figure 4-6 presents a schematic representation of a horizontally emplaced WP with conduits, representing the inlet and outlet. In this schematic, the conduit for liquid entry is shown on the upper half of the WP, and the conduit for the liquid exit is shown on one of the sides. Liquid water will accumulate between the SF rods and in the pore space until its level rises to the level (solid parallelogram) of the exit conduit (dashed parallelogram). Thus, in the bathtub model, the maximum volume of water available for UO<sub>2</sub> matrix dissolution is the level shown by the dotted parallelogram. All SF above this level is assumed to remain dry and contributes to gaseous releases.

When liquid water enters into the WP following its failure, the mass balance model for the radionuclide inventory in liquid water contacting a failed WP is:

$$\frac{\partial m_i}{\partial t} = w_{li}(t) - w_{ci}(t) - w_{di}(t) - m_i \lambda_i + m_{i-1} \lambda_{i-1} \quad (4-24)$$

where

- $m_i$  = amount of radionuclide  $i$  in the WP water at time  $t$  [mol]
- $w_{li}$  = rate of transfer from the solid fuel into the resident water in the WP due to leaching of the SF [mol/yr]
- $w_{ci}$  = rate of advective transfer out of the WP [mol/yr]
- $w_{di}$  = rate of diffusive transfer out of the WP [mol/yr]
- $\lambda_i$  = decay constant of radionuclide  $i$  [1/yr]
- $m_{i-1}$  = amount of the parent at time  $t$  [mol]
- $\lambda_{i-1}$  = decay constant of the parent [1/yr]

The product,  $m_i \lambda_i$ , is the amount lost due to decay, and  $m_{i-1} \lambda_{i-1}$  represents amount generated by the decay of the parent radionuclide.

The advective mass transfer out of the WP can be represented by (Wescott et al., 1995):

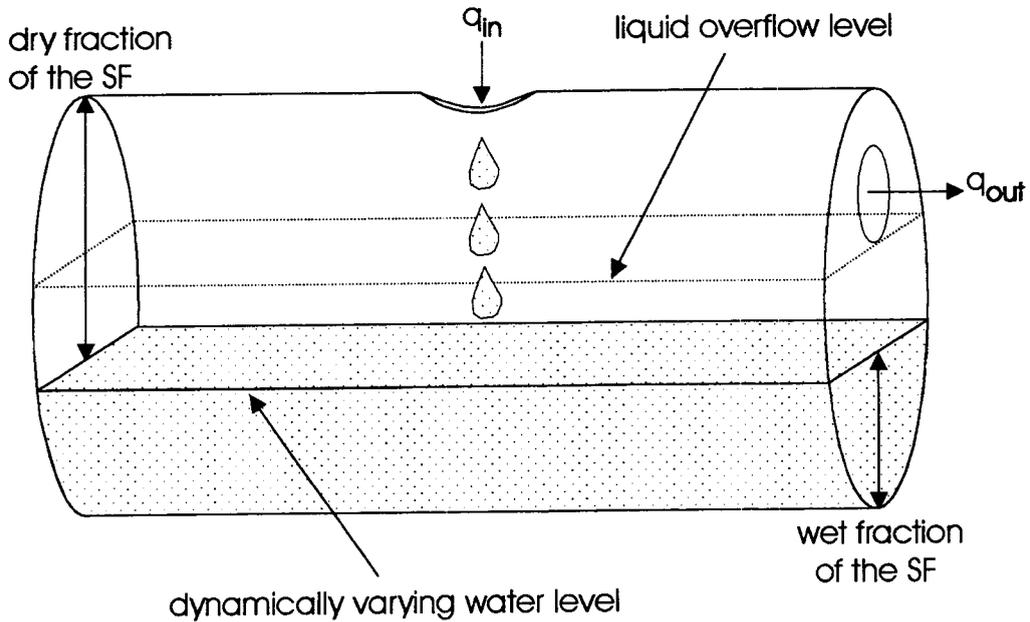


Figure 4-6. Illustration of spent fuel dissolution in a failed waste package

$$w_{ci}(t) = C_i(t) q_{out}(t) \quad (4-25)$$

and

$$q_{out}(t) = q_{in}(t) k_q \left( \frac{V}{V_{max}} \right)^2 \quad (4-26)$$

where

- $C_i$  = concentration of radionuclide  $i$  in the WP water [ $\text{mol}/\text{m}^3$ ]
- $V$  = volume of water in the WP at time  $t$  [ $\text{m}^3$ ]
- $V_{max}$  = maximum volume of water that the WP can hold before water overflow [ $\text{m}^3$ ]
- $k_q$  = weir coefficient [unitless]
- $q_{in}$  = water entering into the WP at time  $t$  [ $\text{m}^3/\text{yr}$ ]
- $q_{out}$  = water leaving the WP at time  $t$  [ $\text{m}^3/\text{yr}$ ]

The Weir coefficient is a damping factor that allows a transient flow system to reach steady state according to its magnitude. Advective mass transfer is considered to occur from the inside to the outside of the WP and instantaneously through a fracture connecting the outside surface of the WP with the EBS-host rock interface, see figure 4-7. Diffusive loss from the fracture (nearly 1.6 m long) is not accounted for in the calculations.

Diffusive mass transfer takes place through the medium surrounding the WP, as shown schematically in figure 4-7. After the water leaves the WP, it is assumed to envelope the whole outer surface of the WP. As a result, the radionuclides are present in uniform concentration. Then, assuming that the WP can be represented by a sphere with equal surface area, the diffusive loss due to the molecular diffusion of a radionuclide into the porous medium can be represented by:

$$w_{di}(t) = 4\pi r^2 \phi D \left. \frac{\partial C}{\partial r} \right|_{r=r_0} \quad (4-27)$$

where

- D = diffusion coefficient of the radionuclide in the medium surrounding the WP [m<sup>2</sup>/yr]
- r = radius of the medium surrounding the WP [m]
- φ = porosity of the medium surrounding the WP [unitless]

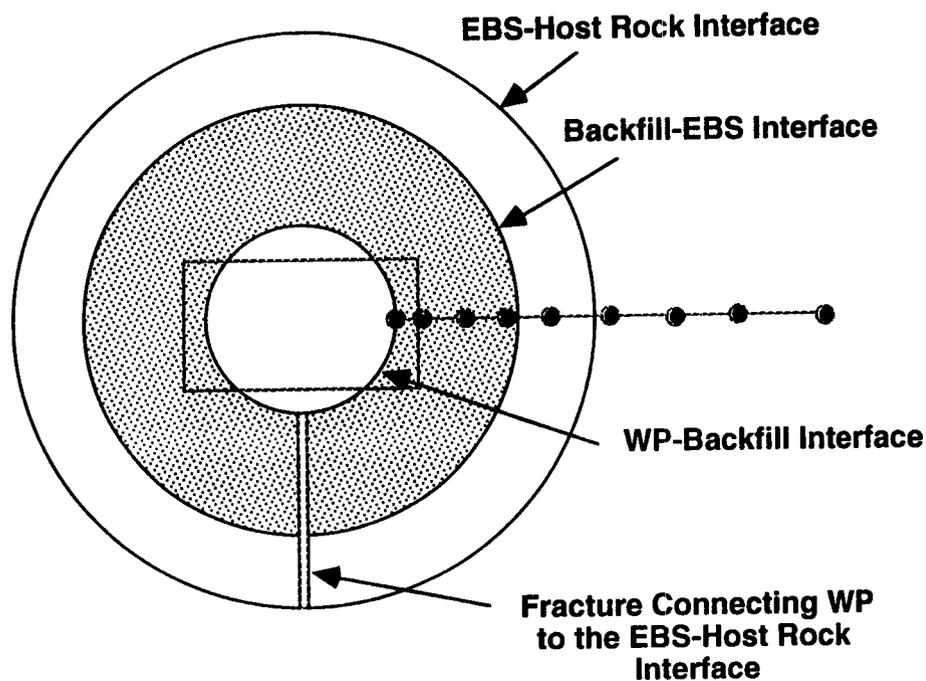
The inventory in the failed WP is monitored by performing material balance calculation at specified time steps. The mass balance calculation includes depletion due to decay, generation of daughter products, and mass depletion from diffusive and advective releases. For the case in which  $w_{li}(t) = w_{ci}(t) = w_{di}(t) = 0$ , Eq. (4-23) reduces to:

$$\frac{\partial m_i}{\partial t} = -m_i \lambda_i + m_{i-1} \lambda_{i-1} \quad (4-28)$$

$$m_i|_{t=0} = m_{i0} \quad (4-29)$$

These differential equations are solved to determine the remaining solid mass of the radionuclides in a chain at a given time  $t$ . The analytical solution (Bateman, 1910) to this initial value problem is given by:

$$N_{ij} = N_{i0} \prod_{k=1}^{j-1} \lambda_k \sum_{l=1}^j \frac{e^{-\lambda_l t}}{\prod_{\substack{m=i \\ m \neq l}}^j (\lambda_m - \lambda_l)} \quad (4-30)$$



**Figure 4-7. Schematic drawing for advective and diffusive mass transfer from the waste package to the host rock**

where

- $N_{ij}$  = contribution from the  $i$ th chain member to the  $j$ th chain member [mol]
- $N_{i0}$  = initial mass of  $i$ th member of chain [mol]
- $\lambda_j$  = decay coefficient of  $j$ th member of the chain [1/yr]

The total amount of the  $j$ th chain member at any time is:

$$N_j = \sum_{i=1}^j N_{ij} \quad (4-31)$$

In the above equations,  $m_i = N_i$  except that  $N_i$  is used to represent radionuclide mass under conditions in which no injection or production of mass occurs, except due to its own decay. In summary, EBSREL provides time dependent release rate in ci/yr to UZFT for all radionuclides and calculates the fractional release rate.

## 4.5 UZFT

The unsaturated zone flow and transport (UZFT) module describes the temporal and spatial variation of deep percolation and radionuclide transport from the repository horizon to the water table. The flow model is based on assuming that gravity drainage occurs in each matrix block, with flow preferentially partitioned into the matrix up to a limiting saturation. Interaction between matrix and fracture is assumed to occur only at hydrostratigraphic interfaces. It is assumed that the flow system is in a quasi-steady state, so that climatic change quickly propagates to depth. Radionuclide transport is simulated using the NEFTRAN II computer program (Olague et al., 1991) and is assumed to occur in 1D flow tubes. The incoming radionuclide concentrations are provided by the EBSREL module. The resulting radionuclide concentrations at the water table are then provided to the SZFT module for the saturated zone transport calculations.

The flow in the unsaturated zone between the repository and the water table is assumed to be primarily in the vertical direction. Thus the flow representation for the unsaturated zone was assumed to be 1D. The hydrogeologic sequences associated with repository subareas define the flowpaths that are to be analyzed. These flowpaths are described as 1D segments because, in part, of the 1D nature of the flow and a need for efficient calculational approaches suitable to the numerous simulations required for sensitivity and uncertainty analyses. Each of the hydrogeologic sequences can be either matrix or fracture flow controlled, but not both. The NEFTRAN II (Olague et al., 1991) computer code was selected to simulate liquid flow and radionuclide transport, because of its relatively high computational speed.

The UZFT module contains a preprocessor which prepares the NEFTRAN II input file. Information for the construction of the NEFTRAN II input file is obtained from other TPA modules, such as UZFLOW. For instance, UZFLOW calculates the percolation fluxes for 1D columns in equilibrium with climatic conditions. In addition, sampled parameter values are obtained from the SAMPLER utility module. The preprocessor in UZFT determines the areal flux, fracture flow, and retardation factors.

UZFT provides the unsaturated flow field in terms of flowpaths (migration legs) and pore velocity. NEFTRAN II is operated in the distributed velocity model (DVM) mode for transport calculations. Convective transport is simulated by moving groups or packets of particles (representing dissolved radionuclides) along the flow field over each time step. Dispersion is simulated by allowing the packets to spread simultaneously with convective transport. Thus, DVM is similar to particle tracking methods or the method-of-characteristics. NEFTRAN II is based upon the following convective-dispersion transport equation in one dimension (Olague et al., 1991):

$$\frac{\partial \rho}{\partial t} = D \frac{\partial^2 \rho}{\partial x^2} - \bar{v} \frac{\partial \rho}{\partial x} + S \quad (4-32)$$

where

- $\rho$  = particle density [mol/m<sup>3</sup>]
- $\bar{v}$  = average particle velocity [m/yr]
- $D$  = dispersion coefficient [m<sup>2</sup>/yr]
- $S$  = volumetric source/sink [mol/(m<sup>3</sup>-yr)]

Figure 4-8 illustrates the use of the NEFTRAN II to represent a 1D unsaturated zone flow path for a subarea of the repository. The UZFT module provides the radionuclide concentrations at the water table for discrete times. This information is then used by the SZFT module for the SZ transport calculations. A more detailed description of the NEFTRAN II program can be found in Olague et al. (1991).

#### 4.6 SZFT

The saturated zone flow and transport (SZFT) module describes radionuclide transport in the SZ, from the location at which radionuclides enter the water table immediately below the repository, to receptor sites in the Amargosa Desert. The SZFT module obtains radionuclide input rates to the water table beneath the repository from the UZFT module and calculates the radionuclide release rates at the compliance boundary which is then used by the DCAGW module. The SZ transport model consists of an array of 1D streamtubes originating at the water table below the repository and terminating at one or more radionuclide receptor locations. Radionuclide transport in the SZFT module is simulated using the NEFTRAN II code (Olague et al., 1991) which calculates the radionuclide groundwater concentration (pCi/L) at the down gradient receptor location.

In the UZFT module, the repository area is divided into a specified number of subareas. In each subarea, a 1D model is used to simulate radionuclide transport from the repository to the water table. In the SZFT module, there are an equal number of SZ streamtubes each connecting to one UZ streamtube. Radionuclides reaching the water table via a 1D unsaturated zone transport column (i.e., UZFT module) are then input to the streamtube defined by the SZFT module and mixed at their point of entry within the streamtube cross section. Figure 4-8 illustrates the method used to represent a streamtube assuming 1D flow in the SZ.

Data and assumptions used in determining the streamtube geometries, seepage velocities, and longitudinal dispersivities are described in Baca et al. (1996). These authors performed a series of 2D computer simulations of groundwater flow and radionuclide transport to assess groundwater dilution and its dependence on the hydrogeologic characteristics of the YM setting. A 2D representation of planar flow from the repository site to the Amargosa Desert (i.e., the potential location of a farmer/rancher critical group) (LaPlante et al., 1995) was considered to assess the extent of hydrodynamic dispersion that may occur as contaminant plumes move through relatively long, heterogeneous flow paths.

Conceptualization of lateral flow used by Baca et al. (1996) was based largely on information from previous DOE modeling studies (Czarnecki and Waddell, 1984; Wilson et al., 1994) and existing field data. The lateral flow model consisted of a 580-km<sup>2</sup> flow tube extending from the repository site south to Amargosa Desert. A subdomain of the Czarnecki and Waddell (1984) regional flow model was selected by tracing selected streamlines west and east of the proposed repository. Locations of the upper and lower boundaries of this streamtube were taken coincident with head contours of 800 and 675 m, respectively, as estimated from available field measurements. The streamtube, shown in figure 4-9, was divided into seven distinct material types or zones. In addition, Baca et al. (1996) investigated the effects of the Ghost Dance and Bow Ridge fault zones on the lateral and vertical flow models.

#### 4.7 DCAGW

The dose conversion analysis for groundwater (DCAGW) module calculates the individual annual total effective dose equivalent (TEDE) from radioactive contamination in the groundwater based on

4-27

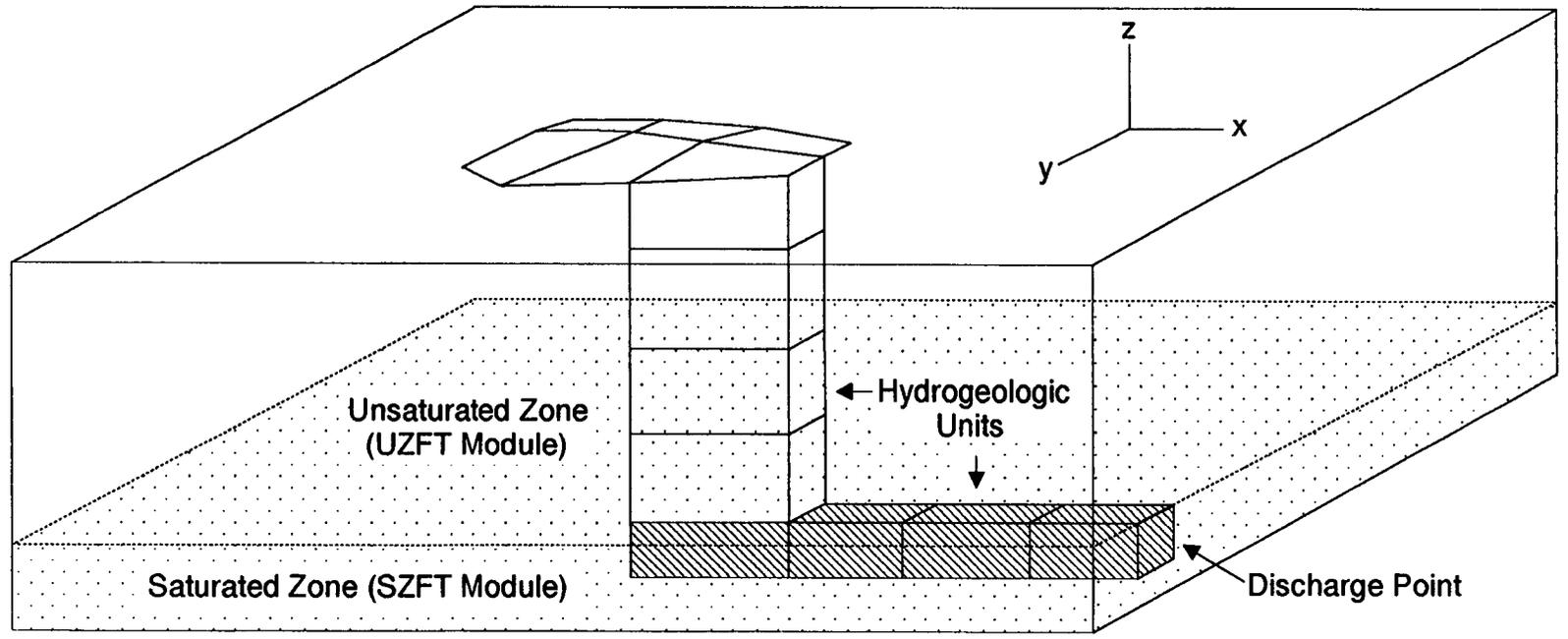


Figure 4-8. Illustration showing both the unsaturated and saturated zone legs for one subarea

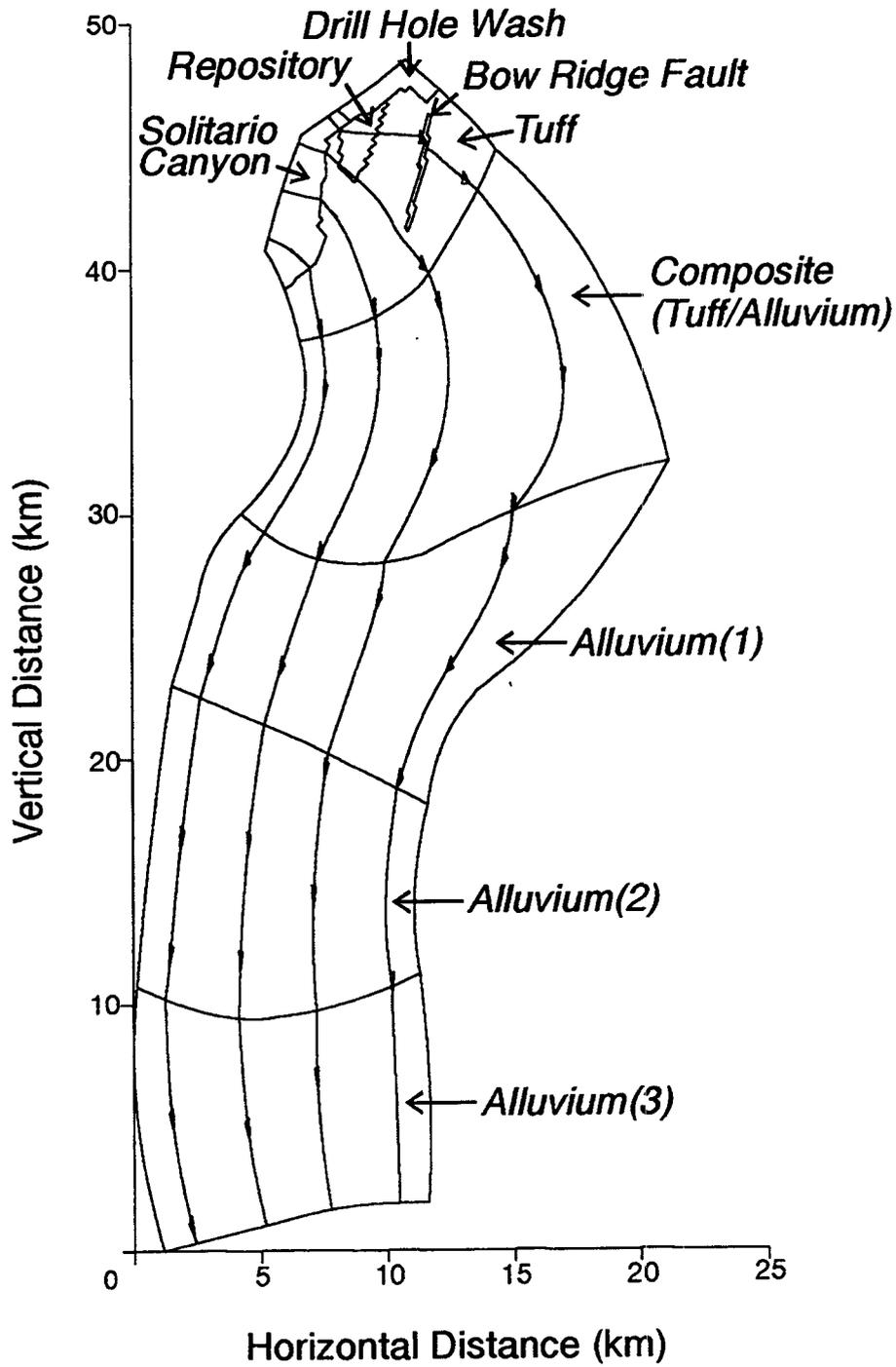


Figure 4-9. Example of streamtube with lateral flow (Baca et al., 1996)

groundwater nuclide concentrations calculated by SZFT. The DCAGW module contains two databases of dose conversion factors (DCFs) designed to calculate the annual TEDE for two different receptors: (i) located less than 20 km from the repository and whose exposure is due solely to the consumption of contaminated groundwater, and (ii) located 20 km or more from the repository and whose exposure is due to consumption of contaminated crops, animal products, and groundwater as well as direct exposure and inhalation. The hypothetical receptor at less than 20 km from the repository is assumed to consume water at a rate of 2 liters per day pumped from a groundwater well. The receptor located 20 km or more from the repository also drinks 2 liters per day but also is assumed to be involved in farming activities. The farmer grows alfalfa, for beef and milk cow feed, grows vegetables, fruits, and grain for personal consumption. Drinking and irrigation water is assumed to be pumped from a groundwater well at the farmer's residence. The exposure pathways considered in the farming scenario are illustrated in figure 4-10. LaPlante et al. (1995) provides a more detailed discussion of the methods and assumptions used to calculate the DCFs for the farming scenario.

The ingestion DCFs are based on published values (U.S. Environmental Protection Agency, 1988) for the selected 43 radionuclides. The calculation of TEDEs for the farming scenario (i.e., located at 20 km or more) is based upon a stochastic simulation performed with the GENII-S code (Leigh et al., 1993; Napier et al., 1988) using unit radionuclide concentrations in the groundwater and sampling 43 biosphere and exposure pathway input parameters. While the NRC has documented acceptable values for generic input parameters (Kennedy and Streng, 1992; Nuclear Regulatory Commission, 1994; Daily et al., 1994), site specific parameters were used where applicable (LaPlante et al., 1995).

Agricultural information (e.g., U.S. Department of Commerce, 1989; Nevada Agricultural Statistics Service, 1988) was collected by LaPlante et al. (1995) for southwestern Nevada. South of YM no farms sell food crops, but some farms raise livestock using both pasture land and feed crops. The predominant livestock in Nye county is beef cattle, with some hogs, chickens, and milk cows. Feed crops are predominantly hay (e.g., alfalfa) with limited amounts of grain. Pasture land is also used for livestock.

Livestock, as well as humans, consume well water. Local water permit information was obtained from the Nevada Division of Water Resources (1995) confirming water use assumptions used in IPA Phase 2 (Wescott et al., 1995) and adding more detailed information of water use in the area. Soil characteristics information for selected farms in the Amargosa Desert area was obtained from local and national offices of the Soil Conservation Service. Resuspension information applicable to the Nevada Test Site provided information for modeling doses from contaminated soil resuspension (Anspaugh et al., 1975; Otis, 1983; Breshears et al., 1989). Crop interception factors were obtained from recent international efforts (International Atomic Energy Agency, 1994) as well as earlier studies (Nuclear Regulatory Commission, 1977; Baes et al., 1982; Hoffman et al., 1982; Snyder et al., 1994).

The unit concentration-based TEDEs (i.e., DCFs) were calculated by execution of GENII-S outside of the TPA code. The arithmetic mean TEDE for each radionuclide formed the basis for the unit groundwater concentration DCFs in the DCAGW module. For each TPA realization, DCAGW multiplies each DCF with the corresponding radionuclide groundwater concentration generated from the SZFT module. For each time-step, the products of each radionuclide concentration and DCF are then summed to calculate a total dose from all radionuclides.

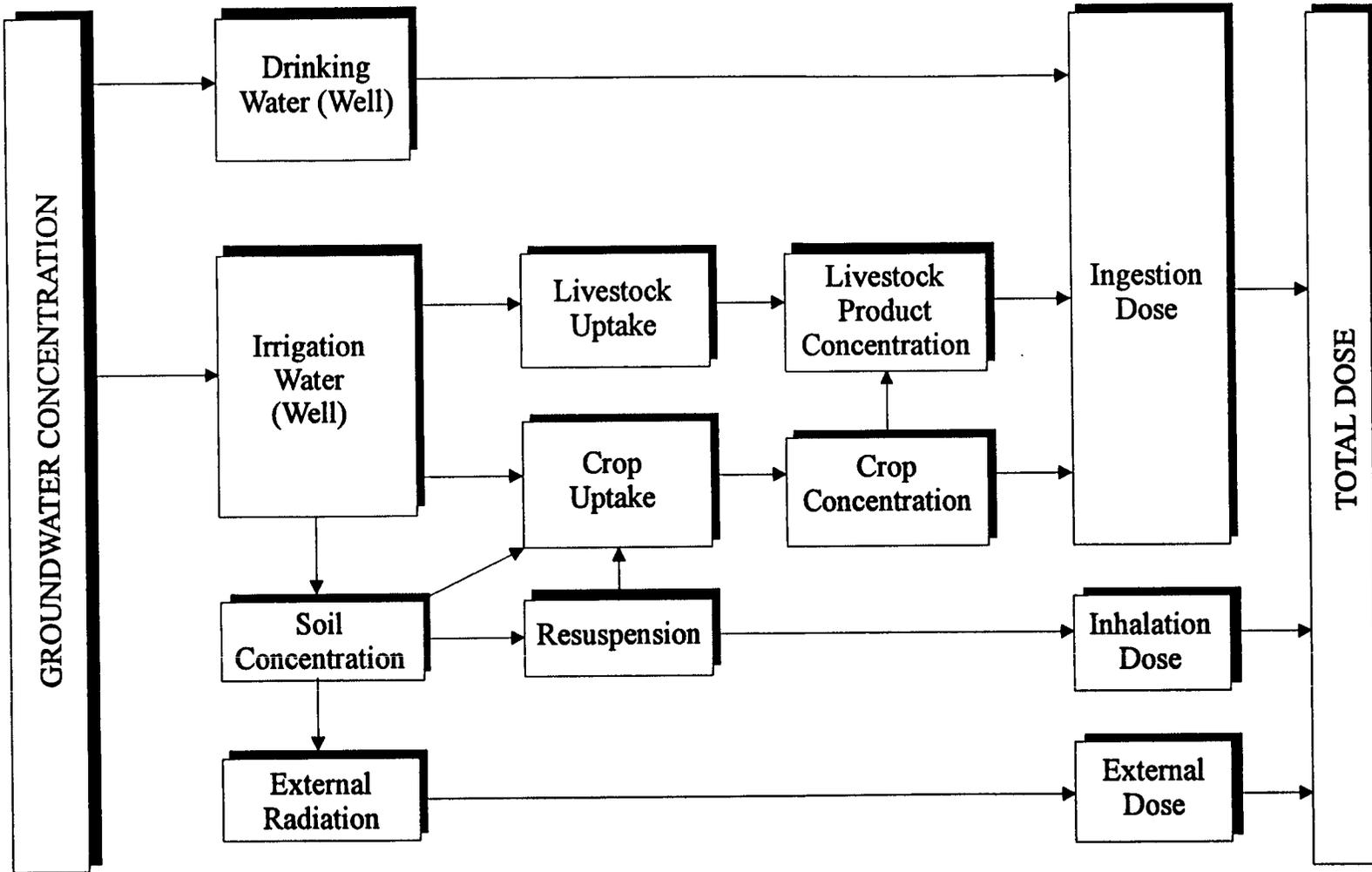


Figure 4-10. Groundwater dose assessment exposure pathways

## 4.8 CLIMATO

The climate (CLIMATO) module is used to analyze effects of climate change on release and transport of radionuclides from the repository. Outputs from CLIMATO are used by UZFLOW as described in section 4.1. A time-series process is used to generate a climatic record with deterministic climate variation sequence and regularly spaced perturbations (e.g., changes from century to century). An input file specifying functions of full-glacial MAP and MAT at particular points in the future is supplied, with enough points to define climatic variation. These values can be based on expert elicitation data (DeWispelare et al., 1993) within 10,000 yr or on Milankovich cycles or Devils Hole data for longer intervals. Linear interpolation of the statistical parameters is used to define the parameters at intermediate times. In CLIMATO, six parameters can be sampled from specified distributions:

- Mean of MAP at full glacial maximum
- Mean of MAT at full glacial maximum
- Standard deviation of MAP within the climate time period
- Standard deviation of MAT within the climate time period
- Correlation between MAP and MAT within the climate time period
- Index into static file containing distributed perturbations

IPA Phase 2 behavior can be modeled by specifying that both standard deviations are zero and providing an input file with all zeros for the fraction of full-glacial maximum.

The time-history values are normalized to the sampled values for full glacial conditions. It is assumed that there is no correlation between perturbations in successive climate time periods. For a time period  $k$ , perturbations are generated using:

$$\chi_{kj} = \rho_{ij} \epsilon_i \quad (4-33)$$

where

- $j$  = MAP or MAT
- $\chi_{kj}$  = a vector of correlated perturbations
- $\epsilon_i$  = a vector of independent normally distributed perturbations
- $\rho_{ij}$  = a correlation matrix

Variables MAP and MAT are calculated from perturbations by using the time-varying mean and standard deviations:

$$v_k = \chi_k \sigma_k + m_k \quad (4-34)$$

where

- $v_k$  = value of MAP or MAT at time  $k$
- $\sigma_k$  = standard deviation of MAP or MAT (constant in time)
- $m_k$  = mean of MAP or MAT at time  $k$

CLIMATO supplies these MAP and MAT values for any point in the TPI to UZFLOW for calculation of MAI rates.

## 4.9 FAULTO

The faulting (FAULTO) module is used for evaluating the potential of direct disruption of WPs due to fault displacement in the proposed repository block at YM. Potential effects of seismic shaking are addressed in a separate module (SEISMO). In this module, faulting is treated as an external event that occurs in a block containing the repository without regard for tectonic mechanisms responsible for driving the faulting process. This module uses published field data to simulate timing and amount of both largest credible and cumulative displacements along existing (but not adequately characterized) faults and new faults within the proposed HLW repository at YM. For a fault displacement, the FAULTO module calculates the percentage of repository area and the number of WPs disrupted and timing of disruption, if it occurs. FAULTO does not evaluate indirect effects of faulting (e.g., possible effects of fault displacement on groundwater hydrology and flow pathways).

While it is reasonable to assume that WPs will be emplaced in the potential repository in accordance with a prescribed setback distance from known and well-characterized faults, there are uncertainties related to consequences of displacement along yet unknown fault zones (including faults not distinguished or adequately characterized, and possible new faults). Considering the complex nature of faults mapped in the proposed repository block (Spengler et al., 1994) relative to possible widths of the fault zones, occurrence of multiple slip surfaces, and lack of data on amount and timing of displacement, it may be difficult to distinguish and adequately characterize a wide fault zone transecting volcanic rock units. If fault zones penetrating subsurface excavations is not adequately characterized, then these zones may not be recognized and an appropriate setback may not be applied. It is also uncertain whether new faults may develop over the TPI. The FAULTO module provides a tool for evaluating the potential consequences of fault displacement in the proposed repository block and analyzing sensitivity of faulting consequence to uncertainties of input parameters.

Fault displacement is generated in FAULTO along a located fault zone inside the simulation area. These randomly generated fault zones represent those that are known to exist but are not adequately characterized, as well as new faults that may develop in the future during the time frame of regulatory interest.

Because of the abundance of faults in and around the potential repository site at YM (Scott and Bonk, 1984; Scott, 1990; Spengler et al., 1994), and evidence of Quaternary displacement on many of these faults, faulting processes are potentially important considerations for the performance of the proposed repository. Faults in and around the proposed repository site at YM constitute two dominant sets, one consisting of northwest-trending faults and the other consisting of north-northeast-trending faults. Faults in the north-northeast-trending fault system in the vicinity of YM have long been interpreted to exhibit Quaternary displacement (Swadley et al., 1984). Northwest-trending faults have also been mapped in and north of the repository block (Scott and Bonk, 1984; Scott, 1990; Spengler et al., 1994).

It is assumed that the computer generated fault zones possess attributes similar to those of Ghost Dance and Sundance Faults, which have been mapped in the proposed repository block. However, because PDFs are used to generate the characteristics of faulting, these values can be easily modified to incorporate new values as detailed mapping of the repository block becomes available and accepted. Strike direction

is either northwest or north-northeast commensurate with the frequency and orientation of mapped fault trace orientations observed in the field at and near YM. Considering the relatively small diameter of emplacement drifts (~5 m) and the steep (60–90°) dip of most near-surface faults at YM, it is assumed that local variation in fault dip has little influence on number of WPs disrupted. In this sense, the model is 2D (plan view) in the plane of emplacement horizon. Whether a fault intersects the potential repository depends on its location, orientation, and trace length within the simulation area.

Based on published field data, the following variables for the fault zone are selected randomly from ranges of possible values represented as PDFs. The sampled variables for FAULTO are location, orientation, geometry (fault length, dip, and width), fault activity, time of largest credible event, cumulative displacement, magnitude of largest credible displacement, and rate of cumulative displacement. The possibility WP disruption depends upon fault displacement exceeding a threshold value governed by repository and WP design and WP emplacement geometry. If the threshold displacement is exceeded by either largest credible displacement in a single event or by cumulative displacement due to fault creep, the number and locations of WPs intersected and disrupted are calculated based on length of intersection of the fault zone within the repository and corresponding fault zone width.

Preliminary field data from Ghost Dance and Sundance Faults provide the information base for describing faulting events in the vicinity of the proposed repository. Data uncertainty is represented using probability distributions. The following variables are considered for describing faults and faulting events in the proposed repository block (Ghosh et al., 1997):

- Fault zone location
- Fault zone trace orientation (north-northeast or northwest)
- Fault zone geometry (i.e., strike, dip, trace length, width, and number and location of multiple slip surfaces)
- Fault activity (active or inactive)
- Number of largest credible displacement faulting events over TPI
- Time of occurrence of largest credible displacement faulting events
- Amount of largest credible displacement per faulting event
- Amount of cumulative displacement
- Time cumulative displacement exceeds threshold displacement

A threshold fault displacement is specified by the user, and if the fault displacement exceeds this threshold then the packages are assumed to fail.

#### **4.10 SEISMO**

The seismic (SEISMO) module calculates WP disruptions caused by repeated seismic motion. The module predicts seismic events that lead to rock fall onto WPs, which causes stress and deformation of the WP. SEISMO estimates effects from comparatively small magnitude repeated seismic motions and less frequent large magnitude earthquakes.

SEISMO requires a “history” of seismic events over the TPI. Therefore, the seismic history will have numerous events of relatively small magnitude (<0.4 g) and a lesser number of larger magnitude (>0.4 g) earthquakes. The frequency of seismic events is correlated with the magnitude of the events,

typically by using a seismic hazard curve. The hazard curve gives the annual probability (likewise the return period) for events larger than a given magnitude.

For use in the TPA code, the seismic hazard curve is discretized into groups with increasing return period. For example, one discretization of a hazard curve may yield three levels for 0.1, 0.3 and 0.6 g events that have return periods of 100, 1,000, and 10,000 yr, respectively. The history of seismic events is generated by sampling from an exponential distribution using the largest annual probability level (e.g., 100-yr return period has an annual probability of 0.01/yr). This value yields the time of the first event. The type of event (i.e., 0.1, 0.3 or 0.6 g magnitude) is then determined. The key to determining event magnitude is understanding that the seismic hazard curve is cumulative. The return period of 100 yr is for events of 0.1 g or larger magnitude. Hence, it includes the 0.3 and 0.6 g magnitude events. The probability of the event being of magnitude 0.1, 0.3 or 0.6 g is based on the return periods of each. The probability that the event is 0.6 g will be  $100/10,000$ , the probability that the event is 0.3 g will be  $(100/1,000 - 100/10,000)$ , and the probability that the event will be 0.1 g is  $(1 - 100/1,000 - 100/10,000)$ . This probability is conditioned on the fact that the event is of magnitude 0.1 g or greater.

The process is repeated to generate a full history of events. The next event is independent of previous events, and the exponential distribution is used again to get the time between events. The total time to the next event is the sum of the time of the last event plus the time lag between events. This process is continued until the time of the event is beyond the TPI. This method was used by Ahola et al. (1995). The SEISMO module does not perform the stochastic generation of these seismic events. Instead the generation is controlled by the EXEC module and performed with the SAMPLER utility module, which performs sampling of all parameters.

Using the seismic event history developed as described above, SEISMO determines effects on WPs emplaced in the drift assuming:

- Emplacement drift is unbackfilled
- Dynamic vibration of the WP and its support system is negligible
- Thermally weakened rocks of the emplacement drift roof, once loosened by seismic shaking fall due to only gravity
- The surface of the rock falling on the WP is flat

Rocks falling from the roof of an emplacement drift can strike WPs. The dynamic or impact force as a result of this strike is approximated using the principle of conservation of energy assuming the following idealized assumptions (Popov, 1970):

- A WP can be treated as an equivalent linear elastic spring with a spring constant  $k$
- No energy dissipation takes place at the point of impact due to local inelastic deformation of the WP material
- The deformation of WPs is directly proportional to the magnitude of the impact force
- The inertia of the WP resisting an impact may be neglected.

It is assumed that a WP can be treated as an equivalent spring with a spring constant  $k$ . The spring constant,  $k$ , is defined as the force required to deflect or deform the spring one unit. When this spring is subjected to a static weight of  $W$  [ $\text{kg m/s}^2$ ], the static deflection/deformation of this spring can be determined as:

$$\Delta_{st} = \frac{W}{k} \quad (4-35)$$

where  $\Delta_{st}$  [m] is the spring deflection/deformation. Similarly, the maximum impact deflection/deformation,  $\Delta_I$ , [m] is:

$$\Delta_I = \frac{P_I}{k} \quad (4-36)$$

where  $P_I$  is the impact force experienced by the spring. By replacing the spring constant in Eq. (4-35) with the spring constant in Eq. (4-34), the impact force can be related to the weight of the falling rocks:

$$P_I = \left( \frac{\Delta_I}{\Delta_{st}} \right) W \quad (4-37)$$

At the instant the WP is deflected/deformed to its maximum amount, all energy of the falling weight is transformed into strain energy of the WP, and can be represented by:

$$W(h + \Delta_I) = \frac{1}{2} P_I \Delta_I \quad (4-38)$$

where  $h$  is the total falling distance [m]. With further manipulation the effective impact force  $P_I$  is:

$$P_I = W \left[ 1 + \left( 1 + \frac{2hk}{W} \right)^{\frac{1}{2}} \right] \quad (4-39)$$

The weight of rock,  $W$ , falling on the WP will be estimated from the results of a drift stability analysis using the computer code UDEC (Itasca Consulting Group, Inc. 1996). The effects of this impact force on the WP deformation  $\alpha$  and stress  $q_0$  are calculated, as follows (Timoshenko and Goodier, 1987):

$$\alpha = \sqrt[3]{\frac{9\pi^2 P_I^2 (k_1 + k_2)^2}{16 R_1}} \quad (4-40)$$

where

- $\alpha$  = a measure of WP deformation [m]
- $q_0$  = stress on WP [MPa]

$$q_o = \frac{3}{2\pi} \sqrt[3]{\frac{16P_1}{9\pi^2} \frac{1}{(k_1 + k_2)^2 R_1^2}} \quad (4-41)$$

- $R_1$  = radius of WP [m]  
 $k_1$  =  $\frac{1-\mu_1^2}{\pi E_1}$  for WP [1/MPa]  
 $k_2$  =  $\frac{1-\mu_2^2}{\pi E_2}$  for falling rock [1/MPa]  
 $E_1$  = modulus of elasticity of WP [MPa]  
 $\mu_1$  = Poisson's ratio of WP [unitless]  
 $E_2$  = modulus of elasticity of falling rock [MPa]  
 $\mu_2$  = Poisson's ratio of falling rock [unitless]

Equations (4-39) and (4-40) are used to estimate the percentage of WP failure at a given time.

SEISMO provides the fraction of packages affected and timing of WP disruptions to the EXEC which, through the use of several other modules described herein, calculates the release inventory, transport, and dose from WP disruptions.

#### 4.11 VOLCANO

The volcanic (VOLCANO) module provides an estimate of the amount of waste entrained during a volcanic eruption and available for transport to the surface. This estimate is based, in turn, on estimates of: (i) the probability of volcanic eruptions within a subregion encompassing the proposed repository, (ii) dike length and orientation, (iii) area disrupted during flow of magma through a conduit, and (iv) distribution of WPs in the repository. The VOLCANO module uses sampled parameters to simulate the locations of volcanic events within a subregion including the repository. The output of the VOLCANO module consists of the amount of waste available for transport to the surface. This output from VOLCANO module is used by the ASHPLUMO module to estimate disposal and deposition of ash and incorporated SF over the land surface.

Volcanic hazards at the YM site are related to the proximity of YM to small-volume basaltic cinder cones. In general, cinder cones form during single episodes of activity and erupt over relatively brief periods of time, on the order of days to years. Renewed volcanism normally involves the formation of new cinder cones. Over time, this type of volcanic activity results in dispersed cinder cones which comprise a volcanic field. In general, volcanic fields are characterized by relatively low recurrence rates of volcanic activity compared with large central volcanoes, such as composite cones and calderas, but are also often characterized by greater longevity of the magmatic system. The most active basaltic volcanic fields in continental settings have recurrence rates of volcanism on the order of hundreds of years. However, most volcanic fields are characterized by recurrence period on the order of  $10^3$ - $10^6$  yr. Individual volcanic fields may remain active for periods of  $10^6$ - $10^7$  yr and eventually consist of tens to

hundreds of cinder cones and their associated lava flows. Thus, the primary hazard associated with volcanism in the YMR is related to the formation of a new volcanic center, a style of activity typical for the region during the last 10 Ma, rather than reactivation of a pre-existing volcanic center.

Cinder cones are the surface manifestations of igneous activity. At depth, basaltic igneous activity is characterized by dike injection and the formation of conduits that are roughly circular to oblate in shape. Mapping in eroded volcanic fields indicates that dike injection always accompanies cinder cone formation. Dikes are roughly planar features, usually 1 to 5 m in width and several kilometers in length. At shallow depths, dikes are usually found in dike zones or dike swarms consisting of numerous closely spaced dikes injected during a single episode of volcanism (Delaney and Gartner, 1995). Flow generally localizes along a short segment of a dike or dike swarm as an eruption proceeds. This localization leads to the formation of a conduit, generally less than 100 m in diameter, through which magma flow is sustained for the duration of the eruption (Delaney and Gartner, 1995; Hill, 1996). This localization may occur anywhere along the length of the dike and at several locations, as observed during past eruptions at Parícutin, Mexico, Jorullo, Mexico, and Tolbachik, Russia.

The goal of the VOLCANO module is to abstract this complex geometry in a simple form that captures the essential aspects of this type of igneous activity for risk assessment. In this simplified geometry, it is assumed that single dikes occur during volcanic events. The probability of an igneous event is the probability that the center of this dike will fall within a subregion about the repository. This subregion is 36 km<sup>2</sup>. It is assumed that (i) the probability of an igneous event is uniform within the subregion, (ii) volcanic events centered outside this area will not affect repository performance by direct disruption, (iii) the conduit and cinder cone associated with this dike injection form at the center of the dike, and (iv) only one conduit forms with a given dike. The dike length, width, orientation, and diameter of the conduit are based on sampled parameters.

Only extrusive igneous events are considered in the current VOLCANO abstraction because the purpose of the module is to provide a value for the total amount of waste available for transport in the ASHPLUMO module, described in the following section. Secondary effects of volcanism, such as disruption of canisters in sections of the dike far from the conduit, additional thermal loading on canisters due to dike injection, and changes in the level of the groundwater table due to dike injection are not considered in the current VOLCANO module. Coupled processes, such as the occurrence of volcanic earthquake swarms during dike injection (Tokarev, 1983; Yokoyama and De la Cruz-Reyna, 1990; Connor et al., 1993), are not considered in the VOLCANO module.

In VOLCANO, Monte Carlo sampling is used to generate random events corresponding to the center of the dike and the location of the conduit in the rectangular region surrounding (and including) the repository horizon. Estimates of the area of the repository impacted by a dike and the conduit are calculated by the TPA Version 3.0 utility module SUBAREA. From the area of intersection, the number of WPs damaged and the quantities of radionuclides available for transport in the conduit are calculated using the initial inventory of the repository.

VOLCANO simulates a random volcanic event within the subregion. This volcanic event occurs at a time chosen randomly within a specified time period (e.g., between 200 yr and 10,000 yr after closure). The procedure for estimating the locations of igneous events can be summarized as:

- Locate the center of the event within the square subregion, corresponding to the conduit location

- Determine the geometry parameters (e.g., dike length, dike width, dike orientation, and conduit radius)
- Calculate the area in each repository zone overlapped by the dike and conduit and calculate the numbers of WPs affected
- Calculate the MTUs of waste available for transport to the surface

The probability of volcanic events in the subregion of interest and the diameter of magma conduits formed during these events are critical parameters in determining the amount of waste available for transport to the surface. Based on the general characteristics of basaltic volcanic fields in and the site-specific geology of the Yucca Mountain Region (YMR), probability models for volcanic disruption of the candidate repository need to consider several basic features:

- Cinder cones of the YMR are part of a larger magmatic system (Yogodzinski and Smith, 1995; Hill and Connor, 1996). It is unlikely that a major change in the recurrence rate of volcanic activity within this magmatic system will occur during the next million years, or during the performance period for the radioactive waste repository, although basaltic volcanism during this period may be strongly episodic.
- Small-volume basaltic volcanoes are clustered within this magmatic system. These clusters are areas within which volcanism has recurred at much greater than average rates during the last 6–8 Ma. YM is located very close to one of the most active of these clusters: the Amargosa Valley-Crater Flat cluster, which consists of numerous Pliocene and Quaternary cinder cones [Connor and Hill (1995)].
- Volcanic eruptions occurred within Amargosa Valley-Crater Flat cluster at 3–4 Ma, with the eruption of basalts both in Amargosa Valley and Crater Flat, about 1.0 Ma, with the formation of five cinder cones in Crater Flat, and about 0.1 Ma, with the formation of Lathrop Wells cinder cone. This recurrence rate, and comparison with activity in the magmatic system as a whole, provides strong indication that future volcanism can recur in the Amargosa Valley-Crater Flat cluster.
- Individual cinder cones in Crater Flat form vent alignments that are coincident with the trends of mapped faults in the YM block and adjacent areas (Connor et al., 1997). Thus structural control on dike propagation, and regional and local stress orientations, must be considered in estimates of the probability of volcanic disruption (Conway et al., 1997). The occurrence of basaltic ash along shallowly buried fault planes in the YMR provides additional indication of the linkage between structure and volcanic activity in the region.

Numerous studies have been conducted to estimate the probability of future volcanism in the YMR and at the candidate repository site in particular (Crowe et al., 1982; 1992; Ho et al., 1991; Smith et al., 1990; Ho, 1991; Sheridan, 1992; Connor and Hill, 1993, 1995; Geomatrix, 1996). Each of these studies reflects varying assumptions inherent in probability models of volcanic disruption of the proposed repository site, but nearly all rely on the distribution and ages of existing volcanic vents to provide some indication of the recurrence rate.

Crowe et al. (1982) introduced spatially and temporally homogeneous Poisson models to the volcanic hazard assessment at YM, assuming that volcanism is spatially distributed in a uniform random manner over some area. This approach has been adopted in many studies (e.g., Ho et al., 1991; Margulies et al., 1992; Geomatrix, 1996), but the definition and extent of the bounded area within which volcanism may occur vary widely. Smith et al. (1990) and Ho et al. (1991), for example, have suggested that Lathrop Wells cinder cone is the first cone in a developing alignment of volcanic vents. They propose that a conservative model is one in which a narrow bounded area, perhaps as little as 75 km<sup>2</sup>, extending from Lathrop Wells through the repository, parallel to alignments that have formed in the region previously. Assuming a regional recurrence rate of 3–10×10<sup>-6</sup> events/yr, the probability of disruption is on the order of 2–8×10<sup>-7</sup>/yr for a 5.5 km<sup>2</sup> repository design. This probability estimate does not consider dike length in the calculation. Conversely, Crowe et al. (1992) have suggested several models that exclude YM from the area within which volcanism may occur, based on the idea that volcanism is limited to Crater Flat, and a zone extending north of this valley. Thus, homogeneous Poisson models have been used to produce a broad range of probability estimates that depend on the bounded area chosen and, to a lesser extent, the assumed regional recurrence rate of volcanism. These source zone models are not readily evaluated in the current formulation of the VOLCANO module because most of the variation in probability estimates in these source zone models occurs across boundaries that are generally drawn within the 35 km<sup>2</sup> subregion (e.g., Geomatrix, 1996). Under these circumstances, the assumption that probability is uniform across the subregion is not appropriate.

Connor and Hill (1993; 1995) used three nonparametric models to estimate the probability of a volcanic disruption of the proposed repository site. These models include (i) a spatio-temporal nearest neighbor model that treats volcanism as a marked point pattern process, (ii) a kernel method, by Lutz and Gutmann (1995) that treats volcanism as a point process within a specified spatial and temporal bandwidth, and (iii) a hybrid nearest neighbor model that combines aspects of the other two models. Condit and Connor (1996) have tested the nearest-neighbor model using data from the much more active Springerville Volcanic Field, Arizona, and found that application of this spatio-temporal model was more successful at forecasting the locations of subsequent volcanic events than using average recurrence rates, because of the clustered nature of small-volume basaltic volcanism.

The probability that the center of a volcanic event occurs within the 36 km<sup>2</sup> subregion given a volcanic event in the map area, is 0.02. For the full range of models reported in Connor and Hill (1995), the probability of a volcanic event within the subregion, given a volcanic event within the map region, ranges from about 0.01 to 0.025. Connor et al. (1996) and Hill et al. (1996) discussed several ways of incorporating structural control on volcanic events directly into probability models and concluded that incorporating structure tends to increase probability slightly. For these structural models, the probability of a volcanic event within the subregion, given volcanism in the map region, ranges from 0.01 to 0.04.

Ranges of recurrence rates of volcanism in the YMR have been discussed in Ho (1991), Connor and Hill (1993) and Geomatrix (1996). These ranges are generally reported to be between about 4 volcanoes per million years and 10 volcanoes per million years. Hidden events, such as unrecognized volcanic eruptions or dikes without an extrusive component, are not considered in this recurrence rate. This range indicates that the recurrence rate of a volcanic event centered within the subregion is between 4×10<sup>-8</sup>/yr and 4×10<sup>-7</sup>/yr. In some calculations, features like the 12-km alignment of five Quaternary cones in Crater Flat are considered to be one event. This consolidation results in a lower recurrence rate and greater dike lengths, with more than one extrusive event associated with an individual volcanic event. Because these alignments are long, consideration of this type of event would require a larger subregion than 36 km<sup>2</sup>.

Determining the disrupted subsurface for a basaltic volcano is an important parameter in calculating the amount of high-level radioactive waste that can potentially be transported into the accessible environment (Hill, 1996). Although there is evidence that some Quaternary YMR volcanoes disrupted anomalously large amounts of subsurface rock, the volume of disruption cannot be calculated directly. An analog volcanic eruption at Tolbachik volcano in Kamchatka, however, provides critical constraints on the subsurface area potentially disrupted by basaltic cinder cones. During a 12-h period of the 1975 Tolbachik eruption,  $2.8 \times 10^6$  m<sup>3</sup> of subsurface rock was brecciated and ejected from the volcano. Using geologic and geometric constraints, this volume represents a circular diameter of disruption of  $49 \pm 7$  m.

An unusual type of volcanic bomb was produced during this disruptive event, which contains multiple subsurface rock types that each show a range in thermal effects. These bombs represent the mixing of different stratigraphic levels and significant distances outward from the conduit during a single event. The same type of unusual volcanic bombs are found at Lathrop Wells and Little Black Peak volcanoes in the YMR. In addition, Lathrop Wells and, to a lesser extent, Little Black Peak have anomalously high subsurface rock-fragment abundances and are constructed of loose, broken tephra. These characteristics strongly suggest that subsurface disruptive events analogous to that at 1975 Tolbachik occurred at these YMR volcanoes. Rock fragment types, stratigraphic relationships, and geophysical data indicate Lathrop Wells disrupted a 0.5–2-km-thick crustal section, which is comparable to the thickness disrupted at 1975 Tolbachik. The volume of Lathrop Wells is nearly identical to 1975 Tolbachik.

Because the volume of disrupted subsurface rock is probably directly related to eruption volume, Lathrop Wells likely disrupted a subsurface volume similar to 1975 Tolbachik. Thus, the subsurface conduits of future YMR basaltic eruptions may have the potential to widen on the order of  $49 \pm 7$  m in diameter. Conduit responses to stress anisotropy in the disturbed geologic setting of the proposed repository site may result in an ellipsoidal cross-sectional area with major axis lengths greater than  $49 \pm 7$  m.

#### 4.12 ASHPLUMO

The ashplume dispersion (ASHPLUMO) module is used to evaluate consequences of extrusive volcanic events in the vicinity of YM. Specifically, ASHPLUMO calls the stand-alone program ASHPLUME (Jarzemba et al., 1997) that evaluates dispersion and deposition of ash and incorporated SF particles resulting from these events. The ASHPLUME code calculates the areal density of SF (in grams of SF per cm<sup>2</sup>) at points on the surface of the earth after an extrusive volcanic event penetrates the repository and exhumes SF. Using published data for wind velocity at the YM site and the estimate of pertinent volcanic parameters of events similar to those that may have occurred at YM in the past, the ASHPLUME code simulates the transport of contaminated particles (composed of SF and ash) to surface points downwind. The SF concentration from deposition can then be used to calculate the dose to members of the local population.

Basaltic volcanism can encompass a variety of eruption styles, depending on the eruption energy. The energy of basaltic eruptions varies from effusive activity, where the predominant product is lava flows, to explosive activity, resulting in fragmentation of the magma into scoria fragments and transport of scoria in the atmosphere as pyroclasts. This latter style of activity generally results in the formation of cinder cones, such as those found in the YMR. Explosive volcanic activity of this kind has the potential to cause dispersal of radionuclides through the biosphere. This dispersion of radionuclides resulting from

volcanic activity is modeled in the ASHPLUME code using approaches originally developed to model the dispersal of ash after volcanic eruptions (Suzuki, 1983).

Figure 4-11 shows the exposure scenario investigated with the ASHPLUME code. The exposure scenario analyzed can be divided into four subprocesses:

- (i) magma enters the repository and becomes contaminated with SF particles
- (ii) tephra forms from the magma and SF is incorporated into tephra (Jarzemba and LaPlante, 1996)
- (iii) eruption column and contaminant plume form and produce fallout at various distances downwind from the volcano (Suzuki, 1983; Jarzemba, 1996)
- (iv) radionuclide contamination causes doses to be incurred at receptor locations (LaPlante et al., 1995; Jarzemba and LaPlante, 1996)

To assess the hypothetical radiation doses that would occur after a basaltic eruption, the distribution of radionuclides in the biosphere after such an event needs to be estimated. It is assumed that the ash particles from the eruption are the carrier of the radionuclides. Methods used previously to estimate radionuclide dispersal by volcanism (Wescott, et al., 1995) theorize that the ash cloud travels in a Gaussian plume, released at a stack height of one half of the volcanic column height. Application of the Gaussian plume model presumes that a plume of contaminants travels in the same direction as the prevailing wind (x direction), but may be somewhat depressed toward the earth's surface due to gravitational settling. Contaminants in the plume follow a Gaussian distribution in the dimensions perpendicular to the direction of travel (y and z directions).

The Gaussian plume model is suitable for modeling airborne and ground concentrations of contaminants for a point source release of contaminants above the surface of the earth (i.e., the stack height). A point source approximation may not be appropriate for a volcanic eruption because a volcanic eruption column is a line source of contaminants in the upward direction. Also, the Gaussian plume model does not accurately account for the effects of gravitational settling of volcanic particles with large diameters (i.e., centimeters). This shortcoming may lead to the gaussian plume model predicting much greater particle ranges than would be the case in reality and hence wider radionuclide distributions than would normally be expected after a basaltic eruption. This wider distribution of radionuclides may tend to underestimate the radiation exposure of persons in a critical group. The critical group is defined as a small, homogenous group (generally ones to several tens of people) who are at the highest risk of incurring additional health effects from the proposed repository. This approach has recently been recommended as the standard of measuring compliance for YM (National Academy of Sciences, 1995).

Models to predict the distribution of ash after an eruption have been developed with the intention of relating eruption magnitude to ash dispersion (Suzuki, 1983; Hopkins and Bridgeman, 1985; Glaze and Self, 1991). The ASHPLUME code uses the model described in Suzuki (1983) that relates eruption magnitude to ash distribution, which is modified to relate eruption magnitude to SF distribution for YM based on a few simple assumptions. The model uses Monte Carlo sampling to determine the power and duration of the eruption, along with other properties of the ash particulates, and develops a SF distribution from those sampled parameters. The SF distribution can be translated into the radionuclide distribution which can then be used to model dose to man.

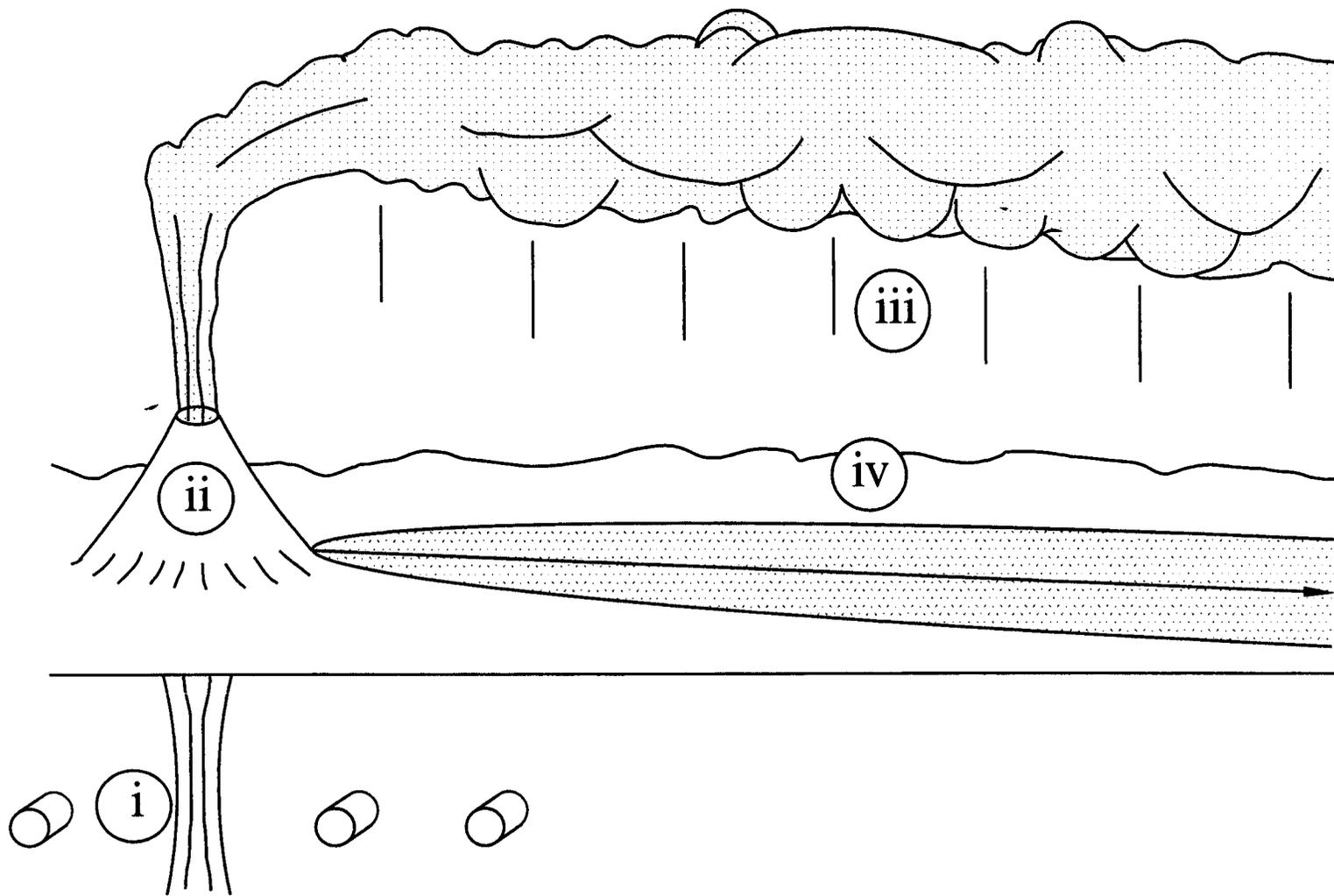


Figure 4-11. Volcanic scenario implemented in the ASHPLUMO module

The model described in Suzuki (1983) can be summarized by the equation that describes the areal density of accumulated ash on the earth surface after an eruption:

$$X(x,y) = \int_{\rho=\rho_{\min}}^{\rho_{\max}} \int_{z=0}^H \frac{5QP(z)f(\rho)}{8\pi C(t+t_s)^{5/2}} \exp\left[\frac{5\{(x-ut)^2+y^2\}}{8C(t+t_s)^{5/2}}\right] d\rho dz \quad (4-42)$$

where

- $X(x,y)$  = mass of ash per unit area accumulated at location  $(x,y)$  [ $g/cm^2$ ]
- $\rho$  = common logarithm of particle diameter  $d$  [where  $d$  is in  $cm$ ]
- $\rho_{\min}$  = minimum value of  $\rho$
- $\rho_{\max}$  = maximum value of  $\rho$
- $z$  = vertical distance from the ground surface [ $km$ ]
- $H$  = height of the eruption column above the vent [ $km$ ]
- $x$  =  $x$  coordinate on the surface of the earth [ $cm$ ]; coordinate oriented in the same direction as the prevailing wind
- $y$  =  $y$  coordinate on the surface of the earth [ $cm$ ]; coordinate oriented perpendicular to the direction of the prevailing wind
- $Q$  = total quantity of erupted material [ $g$ ]
- $P(z)$  = function for particle diffusion at height within  $dz$  about  $z$  [unitless]
- $f(p)$  = distribution function for particles with a log-diameter within  $dp$  about  $p$  normalized per unit mass [unitless]
- $C$  = constant relating the eddy diffusivity and the particle fall time [ $cm^2/s^{5/2}$ ]
- $t$  = particle fall time [ $s$ ]
- $t_s$  = particle diffusion time in the eruption column [ $s$ ]
- $u$  = wind speed [ $cm/s$ ]

The assumptions used in Suzuki (1983) to derive Eq. (4-43) are (i) erupted material consists of a finite quantity of volcanic particles, (ii) the distribution of the diameter of the released particles has a single mode, (iii) all particles fall at the terminal velocity and finally accumulate on the ground, and (iv) the particles have a probability to diffuse out of the eruption column during their upward travel in the column. These assumptions are more realistic for modeling volcanic releases of radionuclides than the assumptions used in the Gaussian plume model (i.e., a point source of radionuclides released at a single height above the vent) provided that the ash particles are the carrier media for the released radionuclides.

For calculation of dose, the necessary quantity to track is the mass of SF per unit area as a function of position after ash, released from the eruption that penetrates the proposed repository, settles on the surface of the earth. To calculate this quantity, a model for SF incorporation into ash was created. This model requires the introduction of a new function to determine the mass of fuel per unit mass of

volcanic ash as a function of the log-diameter of the ash after the ash has been contaminated with SF,  $FF(\rho^a)$ . The volcanic ash mass is assumed to be distributed lognormally:

$$f(\rho^a) = \frac{1}{\sqrt{2\pi}\sigma_d} \exp\left(-\frac{(\rho^a - \rho_{\text{mean}}^a)^2}{2\sigma_d^2}\right) \quad (4-43)$$

where

- $\rho^a$  = log-diameter of ash particle size [particle size in cm]
- $\rho_{\text{mean}}^a$  = mean of the log-diameter of ash particle size [particle size in cm]
- $\sigma_d$  = standard deviation of the log particle size
- $f(\rho^a)$  = normalized (per unit mass) probability distribution of ash mass as a function of  $\rho^a$

To determine  $FF(\rho^a)$  the fuel fraction (ratio of fuel mass to ash mass) as a function of  $\rho^a$ , one must consider that all fuel particles of size smaller than  $(\rho^a - \rho_c)$  have the ability to simultaneously be incorporated into volcanic ash particles of size  $\rho^a$  or larger, where  $\rho_c$  is the incorporation ratio. The fuel fraction as a function of  $\rho^a$  is determined by summing all the incremental contributions of fuel mass to the volcanic ash mass from fuel sizes smaller than  $(\rho^a - \rho_c)$ . An expression for the fuel fraction is given as:

$$FF(\rho^a) = \frac{U}{Q} \cdot \int_{\rho=-\infty}^{\rho=\rho^a} \frac{m(\rho - \rho_c)}{1 - F(\rho)} d\rho \quad (4-44)$$

where

- $Q$  = total mass of ash ejected in the event [g]
- $U$  = total mass of fuel ejected in the event [g]
- $F(\rho^a)$  = cumulative distribution of ash mass with  $\rho^a$ ,  $f(\rho^a)$
- $m(\rho - \rho_c)$  = normalized probability distribution of fuel mass with  $(\rho - \rho_c)$

This equation assumes the resulting contaminated particles follow the same size distribution as the original volcanic ash particles. This assumption seems reasonable since, for most events sampled in these analyses, the total mass of volcanic ash is on the order of  $10^{13}$  to  $10^{15}$  g and for most events, only several WPs are disrupted ( $10^7$  g of fuel each). For each simulation, Eq. (4-45) is numerically integrated to calculate the distribution of the SF and volcanic ash on the earth's surface resulting from a basaltic

eruption that is assumed to disrupt the repository. The integrand of Eq. (4-45) is multiplied by  $FF(\rho^a)$  and then recalculated to find the SF density at location  $(x,y)$ .

The model developed by Suzuki (1983) is appropriate for particles of mean diameter greater than about 15 to 30 micrometers. This cutoff is generally accepted to be the lower limit for the importance of gravitational settling of particles (Cember, 1983; Heffter and Stunder, 1993). For particle sizes less than about 15 micrometers, atmospheric turbulence is great enough to keep the particle aloft for a longer time than would be predicted by the model. Since the typical mean diameter of ash particles after an eruption is generally much larger than 15 micrometers (Suzuki, 1983), this model is useful for calculating the distribution for the vast majority of ash and, hence, radionuclides released. Jarzempa and LaPlante (1996), Jarzempa (1996), and Jarzempa et al. (1997) contain more complete descriptions of the derivation, structure, and use of the ASHPLUME code and its algorithms.

### 4.13 ASHRMOVO

The ash removal (ASHRMOVO) module models the time-dependent radionuclide areal densities of contaminated soil surface layers subject to removal by leaching, erosion, and radioactive decay. ASHRMOVO is used in concert with ASHPLUMO, which establishes initial radionuclide areal densities for extrusive volcanic events at the time of the event that intersects the waste repository. ASHRMOVO uses the INVENT utility module to decay the inventory for succeeding times. The subsequent time "history" of radionuclide surficial contamination is converted to dose by the DCAGS module.

ASHRMOVO uses generalized analytical solutions to calculate dynamic serial radioactive decay, including nonradioactive decay losses by leaching or erosion. For the volcanic exposure scenario previously described in ASHPLUMO, ASHRMOVO calculates the time history of radionuclide surficial contamination following the event by using analytical solutions to the following differential equations (i.e., INVENT utility modules):

$$\begin{aligned} \frac{d}{dt}N_i(t) &= \lambda_{i-1}^P N_{i-1}(t) - \lambda_i^P N_i(t) - \lambda_i^L N_i(t) - \lambda_i^B N_i(t) \\ &= \lambda_{i-1}^P N_{i-1}(t) - \lambda_i^T N_i(t) \end{aligned} \quad (4-45)$$

where

$N_i(t)$  = time-dependent areal density of radionuclide  $i$  [mol/m<sup>2</sup>]

$N_{i-1}(t)$  = time-dependent areal radionuclide density of the parent [mol/m<sup>2</sup>]

$\lambda_i^T$  =  $\lambda_i^P + \lambda_i^L + \lambda_i^B$

$\lambda_i^P$  = removal (or generation) of a contaminant through radioactive decay [1/yr]

$\lambda_i^L$  = relative leach rate of radionuclide  $i$  from the ash blanket [1/yr]

- $\lambda^B$  = bulk removal of the blanket due to surface erosion [1/yr]  
 $\lambda_i^T$  = total loss rate of radionuclide i from physical decay, leaching, and surface erosion [1/yr]

The relative leach rate of radionuclide i is based on Eq. 4.6.3 of Napier et al. (1988). The relative leach rate has an upper limit,  $\lambda_i^{Lmax}$ , that is dependent on the solubility limit of the radionuclide and the amount of radionuclide present. When the processes are leach rate limited, the relative leach rate of radionuclide i is given by:

$$\lambda_i^L = \lambda_i^{Lmax} \quad N_i < \frac{S_i * R_a}{\lambda_i^{Lmax}} \quad (4-46)$$

and when they are solubility limited it is given by:

$$\lambda_i^L = \frac{S_i * R_a}{N_i} = \frac{S_i * R_a}{\bar{N}_i} \quad N_i > \frac{S_i * R_a}{\lambda_i^{Lmax}} \quad (4-47)$$

where

- $S_i$  = solubility limit of radionuclide i [mol/m<sup>3</sup>]  
 $R_a$  = areal recharge of water [m<sup>3</sup>/(m<sup>2</sup>-yr)]  
 $\bar{N}_i$  = average radionuclide areal density over the time-step [mol/m<sup>2</sup>]

The bulk removal rate of the blanket,  $\lambda^B$ , is a constant value of 0.001 per yr based on preliminary estimates of ash blanket lifetimes, but may be sampled by specifying a distribution in the *tpa.inp* file.

#### 4.14 DCAGS

The dose conversion analysis for ground surface pathways (DCAGS) module calculates dose to humans from exposure to radionuclides in surface soil. The annual TEDE is calculated for 43 radionuclides for each TPA realization and at each time-step. The module utilizes databases of DCFs that convert unit radionuclide areal densities to TEDE (i.e., rem/yr per pCi/m<sup>2</sup>).

Like the DCAGW module, the DCAGS module is designed to calculate the annual TEDE for two different dose receptors: (i) located at a distance less than 20 km from the proposed repository whose exposure is due solely to inhalation and direct exposure, and (ii) located at a distance 20 km or more whose dose is due to ingestion, inhalation, and direct exposure. At this time, the analyses of direct exposure and inhalation from resuspension of contaminated ash are premature, thus the database of dose conversion factors for the closer-in receptor location is universally zero. For the further-out receptor, dose is dominated primarily by ingestion, thus preliminary estimates for doses due to inhalation and direct

exposure are included in calculating the DCFs. Also like the DCAGW module, the databases contain mean values of 125 GENII-S runs for the 43 important radionuclides. Jarzempa and LaPlante (1996) list those values for the further-out receptor. The characteristics of this receptor are also contained in LaPlante et al. (1995).

For each TPA realization, DCAGS multiplies each DCF with the corresponding radionuclide concentration generated. For each time-step the products of each radionuclide concentration and DCF are then summed to calculate a total dose from all radionuclides.

## 5 INPUT DATA

The TPA Version 3.0 code is executed in file batch mode using one main input file: *tpa.inp*. The TPA code writes output information and simulation results to a set of files for subsequent post-processing (i.e., plotting or importance analyses). The file interfaces are described in the following sections.

### 5.1 INPUT TO TOTAL-SYSTEM PERFORMANCE ASSESSMENT CODE

All the input data for the TPA Version 3 code are contained in the *tpa.inp* file. No other external files or input affect the code calculations. Auxiliary data files are used by some consequence modules; however, these files are static. The *tpa.inp* file contains all the information necessary to describe the scenario and the number of realizations requested as well as all of the sampled parameters. The *tpa.inp* file contains two comment lines that the analysts should use to describe the type of run being performed. These lines are read as Character\*80 and echoed at the top of all output files.

### 5.2 FLOW OF DATA BETWEEN MODULES

EXEC controls data flow by passing data in the subroutine call statement to each module. EXEC does not use common blocks or disk files for data transfer between itself and consequence modules. Within a consequence module, common blocks or files can be used. To ensure a modular design, data is not passed directly between consequence modules. Each module is called only by EXEC and not by other modules. For efficiency and implementation purposes, the modules can consist of more than one subroutine, may call TPA code utility subroutines (e.g., INVENT), or may call stand-alone programs (e.g., NEFTRAN). Individual modules pass information only to EXEC to control the simulation process.

The overall sequence of execution and flow of data is shown in figure 2-2. Here, EXEC starts the simulation by reading the *tpa.inp* file through the READER routine. The READER module is called only once during a run. Having determined the parameters that describe the system, the EXEC continues by calling component-specific consequence modules. Some modules are called only once per realization, while others will be called many times. Modules being called once include SAMPLER, DCAGW and DCAGS. The modules CLIMATO, UZFLOW, NFENV, EBSFAIL, EBSREL, UZFT, and SZFT are called once for each subarea and each realization. If disruptive scenarios are being analyzed, the FAULTO, SEISMO, VOLCANO, ASHPLUMO, ASHRMOVO, and DCAGS modules are called directly by EXEC once during a realization. These disruptive modules are used either to cause earlier failure of the EBS or provide a more direct pathway for radionuclides into the biosphere (e.g., VOLCANO through ASHPLUMO). If desired in the future, disruptive scenarios can be designed or modified to affect groundwater flow and transport of radionuclides.

The computational scheme for the TPA Version 3.0 code is shown in figure 2-3. Each module is called by the EXEC module with some inputs and then returns some outputs to the EXEC through the call statement. Consequence modules do not call each other directly. There is a clear expectation of inputs and outputs between the EXEC and each module. In most cases, the output of one module is used to provide input to the next module. The EXEC has two main loops for the number of realizations and the number of subareas.

## 5.3 DEVELOPMENT OF INPUT FILE *tpa.inp*

The *tpa.inp* file is keyword oriented. A keyword is located on a line by itself. Subsequent lines (one or more) describe the data more fully. There are currently a total of 21 keywords available to the user: (i) title, (ii) iflag, (iii) subarea, (iv) aqueousnuclides, (v) constant, (vi) iconstant, (vii) uniform, (viii) loguniform, (ix) iuniform, (x) normal, (xi) lognormal, (xii) beta, (xiii) logbeta, (xiv) triangular, (xv) logtriangular, (xvi) exponential, (xvii) finiteexponential, (xviii) hazardcurve, (xix) userdistribution, (xx) correlateinputs, and (xxi) endoffile. All keywords are case sensitive and should be lower case. Some of these keywords are optional, and may never be needed in a run, while others are required. Numerous error traps in the READER utility module help detect problems with the input data and provide helpful comments to the analyst. The *tpa.inp* file has all of the description required to execute the TPA code. This file is designed to be the only source of input information. Because duplicate sources of the same data can become a problem in any large TSPA, the *tpa.inp* file is designed to accommodate the wide variety of data needing to be input. Although this file is expected to be the only input data file for the TPA code, other static data files such as the digital elevation maps for the ground surface are more conveniently stored in archived files elsewhere. These archival files represent data that are not changeable for an application.

Comments can be added to the input file for the benefit of the analyst. The comment line begins with the first character being "\*". Comment lines should be placed between groups of input data. For example, three lines may be required to describe one input parameter, and another three for the second parameter. Comment lines should be used after the first parameter is fully described and before the second is described.

A number of the keywords are specific to the EXEC, but most are generic and used by the consequence modules. For example, the title, subarea, and aqueousnuclides are required and used by the EXEC. The other keywords are more generic and used by the consequence modules as well as the EXEC. These keywords typically assign values (or statistical distributions) to parameters. The parameters are introduced in the *tpa.inp* file, expecting to be needed in the EXEC or consequence modules. Each parameter has a name which is of character type and up to length 60. The name of each parameter must be unique otherwise, the SAMPLER will produce an error message and stop execution if a parameter is defined twice. The module developer who introduces the parameter and assigns the name will use the name to query the value from within the code of a consequence module. Hence, the module developer that has introduced the parameter must also query the value using precisely the same name. In this way, the SAMPLER allows the values of parameters to be obtained by consequence modules. Each keyword is described in the following subsections.

### 5.3.1 title

The analyst is required to provide a two-line title that uniquely identifies the simulation case. This requirement is both common analysis practice and provides useful QA documentation. The description should be sufficiently specific to describe the particular problem being analyzed. The format for the title line is the keyword title, a character string of length 5. The next two lines contain the two 80 character long titles. Below is an example of title for a code run:

```
**
title
This is the test file for TPA 3.0 BETA version
which is being supplied to NRC on January 31, 1997.
**
```

The "\*\*\*" indicates a comment line, one above and one below the two title lines. The two title lines cannot be separated by comments.

### 5.3.2 iflag

The keyword iflag is set to either 0 (indicating no or not active) or 1 (indicating yes or active). The flag is used to select an option. For example, the scenarios have been modeled using flags that determine if the disruptive event/process occurs during the time period of interest. For scenarios, this use of flags forces the disruption to occur. The analyst can force the disruption to occur and then weight the predicted consequences by the probability that the disruption occurs. Determining which disruptions will occur and their respective probabilities is performed by the analyst(s) prior to making the runs and external to the TPA Version 3.0 code. This procedure is describe in Chapter 3 of NRC/CNWRA IPA Phase 2 report (Wescott et al., 1995). Below is an example of the use of iflag:

```
**
iflag
Volcanism disruptive scenario flag (yes=1, no=0)
1
**
iflag
Faulting disruptive scenario flag (yes=1, no=0)
1
**
iflag
Seismic disruptive scenario flag (yes=1, no=0)
1
**
```

The above example would run the VOLCANO, FAULTO, and SEISMO consequence modules in the set of runs.

The iflag keyword can also be used to pass information to the EXEC and other consequence modules. This use of iflag makes the most sense when the user has a choice between two options. A good example is the use of the LHS scheme, which is illustrated as shown below:

```
**
iflag
LatinHypercubeSampling (yes=1, no=0)
1
**
```

The EXEC module will check to see if the parameter name "LatinHypercubeSampling (yes=1, no=0)" has been defined in the *tpa.inp* file. If it has, the EXEC will query the integer flag value and it will use LHS if the value is 1 or Monte Carlo random sampling if the value is 0. The above example has been implemented in the default *tpa.inp* file provided to new users of the code. The iflag keyword is also available for the analyst developing a consequence module.

### 5.3.3 subarea

The analyst is required to specify the total number and locations of the repository subareas using the subarea keyword. A subarea is represented by a quadrilateral and defined by specifying the coordinates of the four vertices. An example of a six subarea discretization of the repository is provided as an example (these six subareas are plotted in figure 4-1):

```
**
subarea
6
ZONE T="Upper Block-NW", F=POINT
547615.8      4079673.8
548118.7      4079465.4
548033.3      4079140.2
547460.1      4079328.2
57615.8       4079673.8
ZONE T="Upper Block-NE", F=POINT
548118.7      4079465.4
548621.6      4079257.1
548606.6      4078952.2
548033.3      4079140.2
548118.7      4079465.4
ZONE T="Upper Block-W", F=POINT
547460.1      4079328.2
548033.3      4079140.2
547933.5      4077770.0
547323.9      4077962.2
547460.1      4079328.2
ZONE T="Upper Block-E", F=POINT
548033.3      4079140.2
548606.6      4078952.2
548543.1      4077577.8
547933.5      4077770.0
548033.3      4079140.2
ZONE T="Upper Block-SW", F=POINT
547323.9      4077962.2
547933.5      4077770.0
547997.4      4076342.2
547660.3      4076451.5
547323.9      4077962.2
ZONE T="Upper Block-SE", F=POINT
547933.5      4077770.0
548543.1      4077577.8
548335.5      4076236.8
547997.4      4076342.2
547933.5      4077770.0
**
```

The first line of the example contains the keyword subarea. The second line contains the number of subareas being defined, which affects the number of subsequent lines. Currently, a recommended discretization of six subareas has been provided in the example file shown here. The six subareas overlay the southern most portion of the upper block, which is sufficiently large to accommodate 70,000 MTU of waste at an areal mass loading of 83 MTU/acre. The DOE also has extended the northern boundary of this zone for an additional emplacement area if it is needed. If all this area were filled at 83 MTU/acre, then more than 70,000 MTU of waste would be emplaced. The added space in the northern portion of the repository is not needed if the bottom six subareas are filled at 83 MTU/acre.

Following the number of subareas (six in this example), the location of each subarea is then provided. The format for the subarea vertices is based on the plotting software TECPLOT (Amtec Engineering, Inc., 1993). The next 36 lines of input data are exactly the same lines used in the input file for plotting and visually checking the coordinate locations and repository outline. Although the above format appears awkward, it avoids the error prone process of reformatting or retyping the coordinate data. The format consists of six lines for each subarea. The first and last line are ignored. The first line is specific to the post-processor TECPLOT (defining a zone, with a name, and by points in floating point format). The last line is a repeat of the second line (which closes the box drawn by connecting points). The second through fifth lines define the coordinates of the vertices in Universal Transverse Mercator coordinates. Easting is the first, and Northing is the second coordinate, and both are in units of meters. Another example is provided below:

```

**
subarea
1
Zone T="Upper Block-NW", F=POINT
547615.8  4079673.8
548118.7  4079465.4
548033.3  4079140.2
547460.1  4079328.2
547615.8  4079673.8
**

```

This example defines one rectangular subarea large enough to accommodate 70,000 MTU at 83 MTU/acre. This example is often useful in testing consequence modules, where the looping over the number of subareas is minimized (will do only one loop) before going to the next realization.

### 5.3.4 aqueousnuclides

The analyst is required to specify the number and names of radionuclides to be considered in the groundwater pathway during the run. The TPA code presently supports up to 43 different radionuclides (see section 3.2 on INVENT). The INVENT utility module recognizes the specific nuclides and has stored information for each nuclide such as half-life, specific activity, and possible participation in radionuclide decay chains. As such, the user does not need to provide data on half-life, specific activity, or definition of decay chains. The analyst needs only to specify the number and names of nuclides to be modeled. If the analyst specifies a nuclide not in the list of 43 recognized by INVENT, an error message will be printed and the code will stop. An example of the aqueousnuclides keyword is provided below:

```

**
aqueousnuclides
21
Cm246
U238
Cm245
Am241
Np237
Am243
Pu239
Pu240
U234
Th230
Ra226
Pb210

```

```
Cs137
Cs135
I129
Tc99
Ni59
Cl4
Se79
Nb94
Cl36
**
```

The first line contains the keyword `aqueousnuclides`. The second line contains the number of nuclides being selected. The selection of nuclides to be tracked is not a trivial exercise, and several considerations lead to selecting these 21 [based on Wescott et al. (1995) with the addition of  $^{36}\text{Cl}$  due to its importance in DOE's TSPA-95, e.g. TRW Environmental Safety Systems, Inc. (1995)] from the full set of 43. Unless the analyst has a clear rationale to do otherwise, the default list of 21 nuclides provided in the example input file (shown above) should be used. After the number of nuclides has been established, each nuclide is identified on a separate line. All names are provided in the first six spaces of each line, and are right justified on the sixth space.

### 5.3.5 constant

The `constant` keyword has a wide variety of uses and is one of the simplest inputs. Only three lines are required to define a constant parameter. The first line has the keyword `constant`. The second line has a name for the parameter which is at most of character length 60. The third line has the value of the constant. FORTRAN distinguishes between integer and real types of data, hence the value is a real (floating point) number. The analyst can view the constant as a delta function PDF. In the future, the analyst may want to use a uniform or normal distribution to describe the range of the parameter. It is straight forward to transform a parameter from type `constant` to another type between runs by editing the *tpa.inp* file. Below are two examples of specific constants currently being used:

```
**
constant
ElevationOfRepositoryHorizon[m]
1072.0
**
constant
WastePackagePayload[MTU]
8.8
**
```

The first value is used in the `NFENV` module and shows that the elevation of the upper block of the repository is 1,072 meters above sea level. The second constant is used in the `EBSREL` consequence module and indicates that the average WP has a waste payload of 8.8 MTU.

### 5.3.6 iconstant

The `iconstant` keyword is the same as the `constant` keyword, except that the value must be of type integer, instead of real. Some compilers are more forgiving and will read non integer numbers then convert them into integer values. For example, the SUN FORTRAN compiler (SunSoft, 1996) will read the number 200.5 as 200 and will read the number 2.34e3 as 2340, although neither are originally in integer format. The user is encouraged to only use integers to avoid input problems. The user is also

encouraged to check the output files (especially *cp.tpa* which is described in section 6.3) to verify the data was read properly. Examples of the use of *iconstant* are:

```
**
iconstant
NumberOfRealizations
200
**
iconstant
NumberOfTimeSteps
201
**
```

The first example is currently being used to specify the number of realizations (the number of realizations and number of vectors are synonymous). The second example is used to dictate the number of time steps to be used by the EXEC to synchronize the transfer of data between consequence modules.

### 5.3.7 uniform

The *uniform* keyword specifies the uniform distribution type of PDF for a parameter. This requires three lines consisting of the keyword *uniform*, the unique name of the parameter, and the lower and upper limits of the uniform distribution. The value of the parameter has zero probability outside this range and uniform probability within this range. The TPA code includes many error checking routines such as checking that the minimum value is less than the maximum value. Examples of uniform distributions include:

```
**
uniform
GroundwaterPercolationRate[mm/yr]
0.5, 2.0
**
uniform
WellPumpingRateAtCriticalGroup[gal/day]
1.0e7, 1.0e8
**
```

The first example currently is being used by UZFLOW to vary the ground water percolation rate into the repository horizon. The second example currently is being used in the SZFT module to account for borehole dilution and is used to predict the radionuclide concentration in the well water.

### 5.3.8 loguniform

The *loguniform* keyword starts the definition of a lognormal PDF for a parameter. The full description requires three lines as shown:

```
**
loguniform
name
xlow, xhigh
**
```

The first line contains the keyword `loguniform`. The second contains the unique name of the parameter. The third contains the lower and upper limits of the loguniform distribution. The original values of the lower and upper limits are used, not the log transformed values. As with the uniform PDF, `xlow` must be less than `xhigh`. There is no current example of the use of this distribution in the TPA code, although it was used in IPA Phase 2 [see `qinfil` in appendix A of Wescott et al. (1995)].

### 5.3.9 `iuniform`

The `iuniform` keyword starts the definition of the integer uniform PDF for a parameter. The full description of the parameter and PDF requires three lines. The first line contains the keyword `iuniform`. The second contains the unique name of the parameter, using up to 60 characters. The third line contains two integer values representing the low and high values, inclusively. The low value must be less than the high. Each value beginning with the low and including the high are equally probable. An example of the use is provided below:

```
**
iuniform
RandomNumberToSelect1of125GENIIRealizations
1, 125
**
```

This example was used in an earlier version of DCAGW in which one of the 125 stored results for DCFs is selected and used for the realization.

### 5.3.10 `normal`

The `normal` keyword is used to assign the normal PDF to a sampled parameter. Use of this keyword requires three lines. The first line contains the keyword `normal`. The second line contains the unique name of the parameter, using up to 60 characters. The third line contains two values used to describe the normal PDF. Frequently, analysts use the mean and standard deviation to describe the normal PDF, but not here. For this keyword the normal distribution is described by the value at 3.09 standard deviations below the mean and 3.09 standard deviations above the mean. Another way of describing this distribution is that the corresponding CDF has a value of 0.001 (0.1 percent quantile) and 0.999 (99.9 percent quantile). This method of describing normal distributions is a continuation of that used by Iman and Shortencarier (1984). Below is an example of a normal distribution being assigned to a parameter.

```
**
normal
ThermalConductivityofYMRock [W/(m-K) ]
1.8, 2.2
**
```

This parameter is currently being used by the NFENV module. The thermal conductivity of the rock at YM has a mean of 2.0 with a standard deviation of about 0.0647 W/(m-K).

### 5.3.11 `lognormal`

The `lognormal` keyword starts the definition of the lognormal PDF for a parameter. Use of this keyword requires three lines as shown:

```
**  
lognormal  
name  
v001, v999  
**
```

The first line contains the keyword lognormal. The second line contains the unique name of the parameter, using up to 60 characters. The third line contains two values used to describe the lognormal PDF. The two values represent the parameter values where the CDF has a value of 0.001 (0.1 percent quantile) and 0.999 (99.9 percent quantile). This method of describing lognormal distributions is a continuation of that used by Iman and Shortencarier (1984).

### 5.3.12 beta

The beta keyword starts the definition of the beta PDF for a parameter. This definition requires three lines as shown:

```
**  
beta  
name  
 $\alpha$ ,  $\beta$ , xmin, xmax  
**
```

The first line contains the keyword beta. The second line contains the unique name of the parameter, using up to 60 characters. The third line contains four values used to describe the beta PDF. The first two input values ( $\alpha$ ,  $\beta$ ) control the shape of the beta distribution, and the last two control the range of the parameter. The parameter has zero probability of having a value less than xmin or greater than xmax. Below is an example.

```
**  
beta  
MaxtixPorosityOfTopopahSpringWelded[from pg 7-11 of SNL TSPA-93]  
3.934, 29.567, 0.000, 1.000  
**
```

This example is not currently being employed in the simulations of the proposed repository. It is based on data in the Sandia National Laboratories (SNL) TSPA-93 report (Wilson et al., 1994) for the matrix porosity of Topopah Spring Welded rock unit near the repository horizon.

### 5.3.13 logbeta

The logbeta keyword starts the definition of the logbeta PDF for a parameter. This keyword requires three lines as shown:

```
**  
logbeta  
name  
 $\alpha$ ,  $\beta$ , xmin, xmax  
**
```

The first line contains the keyword `logbeta`. The second line contains the unique name of the parameter, using up to 60 characters. The third line contains four values used to describe the `logbeta` PDF. The first two values ( $\alpha$ ,  $\beta$ ) control the shape of the distribution, and the last two control the range of the parameter. The parameter has zero probability of having a value less than `xmin` or greater than `xmax`. The values of `xmin` and `xmax` are not log transformed values, but the original values. Below is an example.

```
**
logbeta
MaxtixSaturatedHydraulicConductivityOfTSw[from SNL TSPA-93]
0.980, 1.875, 2.88e-13, 1.07e-8
**
```

This example is not currently being used in the simulation of the proposed repository, but is based on data in the SNL TSPA-93 report (Wilson et al., 1994) for the matrix saturated hydraulic conductivity of Topopah Spring Welded rock unit near the repository horizon.

### 5.3.14 triangular

The `triangular` keyword starts the definition of the triangular PDF for a parameter. This description requires three lines. The first line contains the keyword `triangular`. The second line contains the unique name of the parameter, using up to 60 characters. The third line contains three values used to describe the triangular PDF. The values represent the minimum, peak, and maximum parameter values. An example is shown below:

```
**
triangular
SolubilityNp[mole/liter]
5.0e-6, 1.4e-4, 1.0e-2
**
```

This example is being used in the EBSREL module for the solubility of Np. Based on experimental observations, geochemists have estimated the solubility ranges from  $5.0e-6$  to  $1.0e-2$  moles/liter with a peak at about  $1.4e-4$  moles/liter. The distribution is approximated by a triangular distribution.

### 5.3.15 logtriangular

The `logtriangular` keyword starts the definition of the `logtriangular` PDF for a parameter. To use this keyword, the user must specify three lines. The first line contains the keyword `logtriangular`. The second line contains the unique name of the parameter, using up to 60 characters. The third line contains three values used to describe the `logtriangular` PDF. The values represent the minimum, peak, and maximum parameter values. These values are not log transformed. An example is shown below:

```
**
logtriangular
ParticleSizeinAshPlume[mm]
0.5, 55.0, 125.5
**
```

This example is being used in the ASHPLUME module.

### 5.3.16 exponential

The exponential keyword starts the definition of the exponential PDF for a parameter. Specifications for keyword consists of three lines as shown:

```
**  
exponential  
name  
lambda  
**
```

The first line contains the keyword exponential. The second line contains the unique name of the parameter, using up to 60 characters. The third line contains one value used to describe the exponential PDF. The values represent the decay or proportionality constant.

```
**  
exponential  
TimeOfNextVolcanicEventinRegionOfInterest[yr]  
1.0e-8  
**
```

This example could be used to generate the time of the next volcanic event in the region of interest, given its annual probability to be one in a hundred million. The time of the next event ranges from zero to infinity.

### 5.3.17 finiteexponential

The finiteexponential keyword starts the definition of the finiteexponential PDF for a parameter. Three lines, as shown below, are required to use this keyword option:

```
**  
finiteexponential  
name  
lambda, tmin, tmax  
**
```

The first line contains the keyword finiteexponential. The second line contains the unique name of the parameter, using up to 60 characters. The third line contains three values used to describe the finiteexponential PDF. The first value represents the decay or proportionality constant (this is the same for the exponential PDF). The next two values represent the minimum and maximum acceptable time, such that the sampled time will always be between these two values. The finiteexponential has the advantage of forcing an event to occur within a TPI, yet the PDF retains the exponential shape. Below is an example.

```
**  
finiteexponential  
TimeOfNextVolcanicEventinRegionOfInterest[yr]  
1.0e-8, 100.0, 10000.0  
**
```

This example could be used to generate the time of the next volcanic event in the region of interest, given its annual probability to be one in a hundred million and also requiring that the event occur after 100 yr and before 10,000 yr.

### 5.3.18 hazardcurve

The hazardcurve keyword starts the definition of the hazardcurve that is used to generate a time series of events. This description requires at least five lines as shown:

```
**
hazardcurve
name
n
mag(1), period(1)
mag(2), period(2)
.
.
mag(n), period(n)
**
```

The first line contains the keyword hazardcurve. The second line contains the unique name of the parameter, using up to 60 characters. The third line contains one integer value used to describe the number of intervals into which the hazardcurve has been discretized. Following are that same number of lines, each having a magnitude and return period. The only requirements for these values are that the periods are in ascending order, and the magnitudes be monotonic. Below is an example.

```
**
hazardcurve
SeismicHazardCurveforSEISMO
3
0.10  100.0
0.30  1000.0
0.60  10000.0
**
```

This example is used to generate a history of seismic events over the TPI for the SEISMO module. The seismic history will have numerous events of the smallest magnitude (0.1 g) and a few of the higher magnitudes. There are only three magnitudes of events that will be generated in the seismic history (0.1, 0.3, and 0.6 g). The return periods for these events is 100, 1,000, and 10,000 yr.

### 5.3.19 userdistribution

The userdistribution keyword starts the definition of the user supplied discrete PDF. Use of this keyword option requires at least five lines as shown.

```
**
userdistribution
name
n
v(1)
v(2)
.
.
v(3)
v(n)
**
```

The first line contains the keyword `userdistribution`. The second line contains the unique name of the parameter, using up to 60 characters. The third line contains one value used to describe the number of user-supplied values. Each subsequent line contains one of the user-supplied values. The values all have equal probability. Below is an example.

```
**  
userdistribution  
ParticleSizeinAshPlume[mm]  
3  
0.1  
1.0  
10.0  
**
```

In this example, the ash plume particle size is either 0.1, 1.0, or 10.0 mm.

### 5.3.20 correlateinputs

The `correlateinputs` keyword is used to introduce correlations between two input parameters. The correlation is based on the rank transformed parameters. This description requires four lines as shown:

```
**  
correlateinputs  
name1  
name2  
corr  
**
```

The first line contains the keyword `correlateinputs`. The second line contains the unique name of the first parameter, using up to 60 characters. The third line contains the unique name of the second parameter, using up to 60 characters. Both the first and second parameter must be defined elsewhere in the input file. The fourth line contains one value representing the rank correlation between input parameters. Below is an example.

```
**  
correlateinputs  
MatrixPorosityLayer1  
MatrixPermeabilityLayer1[m2]  
0.1  
**
```

This example specifies a rank correlation of 0.1 between the matrix porosity and permeability for the first hydrostratigraphic layer.

### 5.3.21 endoffile

The `endoffile` keyword is used to indicate the last line in the input file. The analyst must specify this keyword once and only once. If it is used more than once, an error message will be printed and the code will stop.

## 6 DESCRIPTION OF OUTPUTS

### 6.1 OVERVIEW

The TPA Version 3.0 code generates a number of output files for later use in sensitivity analyses, evaluation of subsystem performance measures, and total-system performance measures. These output files are summarized in table 6-1. The files, which are in ASCII format, contain data about sampled parameters, constant parameters, module variables, compliance with the EPA release standard, and time-dependent annual doses. The name and a brief description of each file are provided in table 6-1. A more in-depth description of the files is provided later in this section.

The first five lines of each output file are standardized and have the format:

```
title1
title2
code version, time and date of start of run
description of file data, first line
description of file data, second line
```

The first two lines are the title lines defined in the *tpa.inp* file. These lines are of character type and 80 characters in length. The third line contains the name of the code and version of the code being used, followed by the time and data of the start of the run. The fourth and fifth lines contain brief descriptions of the contents of the file. Subsequent lines contain the data. Examples of the output files are presented in the following sections in which file contents are described in more detail.

### 6.2 SAMPLED PARAMETERS

Model input parameters with statistical distributions (i.e., sampled parameters) are designated by the user in the *tpa.inp* file using PDFs such as uniform, normal, lognormal, beta, etc. These parameters are sampled within the SAMPLER utility module once for each realization. The EXEC module writes the values of each sampled parameter for each realization to the file *sp.tpa*. An example of the first few lines of a *sp.tpa* file are shown below:

```
This is the test file for TPA 3.0beta version
which is being supplied to NRC on February 26, 1997.
TPA 3.0, Job started: Fri Feb 28 15:53:25 1997
Sampled Parameters for each realization
Sampled Parameters change each realization
      0          58
GroundwaterPercolationRate[mm/yr]
WellPumpingRateAtCriticalGroup[gal/day]
TimeOfNextFaultingEventInRegionOfInterest[yr]
XLocationOfFaultingEventInRegionOfInterest[m]
YLocationOfFaultingEventInRegionOfInterest[m]
....etc....
      1          58
      0.1448943E+01
      0.6308013E+04
      0.4199900E+04
      0.5496345E+06
      0.4076762E+07
....etc....
```

**Table 6-1. Summary of output files generated by TPA Version 3.0 code**

File Name	Description
sp.tpa	sampled parameters for each realization of the simulation
cp.tpa	constant parameters for the simulation
mv.tpa	module variables (which are scalar) for each realization of the simulation, used to report subsystem performance measures such as waste package lifetime, groundwater travel time, etc.
epaccdf.tpa	EPA-summed normalized cumulative release for the time period of interest (for assessing compliance with the remanded 40 CFR Part 191)
rgwnr.tpa	time-dependent annual effective dose equivalent (EDE) from the groundwater pathway, for each nuclide for each realization
rgwsr.tpa	time-dependent annual EDE from the groundwater pathway, summed over all nuclides for each realization
rgwna.tpa	time-dependent annual EDE from the groundwater pathway, for each nuclide, averaged over all realizations
rgwsa.tpa	time-dependent annual EDE from the groundwater pathway, summed over all nuclides, averaged over all realizations
rgsnr.tpa	time-dependent annual EDE from the ground surface pathway, for each nuclide for each realization
rgssr.tpa	time-dependent annual EDE from the ground surface pathway, summed over all nuclides for each realization
rgsna.tpa	time-dependent annual EDE from the ground surface pathway, for each nuclide, averaged over all realizations
rgssa.tpa	time-dependent annual EDE from the ground surface pathway, summed over all nuclides, averaged over all realizations
rgwgssa.tpa	time-dependent annual EDE from both groundwater and ground surface pathways summed over all nuclides, averaged over all realizations

The first five lines of the example *sp.tpa* are consistent with the general format described previously. The sixth line contains two numbers, the first is the realization count (being initially zero) and the second being the number of sampled parameters. In this example, the number of parameters is 58. Next, the name of each sampled parameter is output, one name per line. These names are the same as given in the *tpa.inp* file. Next is a line having two integer numbers representing the realization number and the number of sampled parameters. The second number is the same as provided prior to the listing of names. It is provided again as a helpful reminder to the user. Next, the numeric information is provided for that parameter for that realization. This is a column of 58 numbers representing a vector of sampled parameters for the first realization. The line with two integers (indicating realization number and total number of sampled parameters) follows with the next vector of sampled parameters, etc. This information is repeated for the total number of realizations.

### 6.3 CONSTANT PARAMETERS

Constant model parameters are specified by the user in the *tpa.inp* file using keywords such as *constant*, *iflag*, and *iconstant*. These parameters are not sampled within the SAMPLER utility module, and the values remain fixed for all realizations in the simulation. The EXEC module writes the values of each constant parameter for the run to the file *cp.tpa*. An example of the first few lines of a *cp.tpa* file are shown below:

```
This is the test file for TPA 3.0beta version
which is being supplied to NRC on February 26, 1997.
TPA 3.0, Job started: Fri Feb 28 15:53:25 1997
Constant Parameters for the run
Constant Parameters do not change during run
    1 = Volcanism disruptive scenario flag (yes=1, no=0)
    1 = Faulting disruptive scenario flag (yes=1, no=0)
    1 = Seismic disruptive scenario flag (yes=1, no=0)
0.188910E+09 = SeedForRandomNumber
    0 = LatinHypercubeSampling (yes=1, no=0)
    10 = NumberOfRealizations
0.100000E+05 = MaximumTime[yr]
```

The first five lines are consistent with the general format described previously. Next, are the values and the names of each of the constant parameters. Both the value and the name are on one line. These names are the same as given in the *tpa.inp* file.

It should be noted that the distinction between a constant parameter and a sampled parameter is helpful for the analyst who desires to perform a sensitivity analysis. The constant parameters have not been included with the truly sampled parameters. If constants were included in the *sp.tpa* file, then its size would be greatly increased. The analyst may change how the parameter is described in the *tpa.inp* file. For example, in one run the parameter may be a constant; in another run, it may be described by a uniform PDF (for example). EXEC would then write the same parameter to either the *cp.tpa* file (if it is a constant) or to the *sp.tpa* (if described by a PDF). By separating the *cp.tpa* and *sp.tpa*, the file sizes are reduced significantly.

## 6.4 MODULE VARIABLES (SUBSYSTEM PERFORMANCE MEASURES)

Module variables are introduced by the code developers and are almost always used to report subsystem performance measures. These variables are identified within the consequence modules and represent important output information useful for sensitivity and importance analyses. The module variables are calculated by the consequence modules for each realization and are stored in the MODULE VARIABLE utility module. The EXEC then prompts the MODULE VARIABLE utility module to print the values at the end of each realization to the file *mv.tpa*. An example of the first few lines of a *mv.tpa* file are shown here:

```
This is the test file for TPA 3.0beta version
which is being supplied to NRC on February 26, 1997.
TPA 3.0, Job started: Fri Feb 28 15:53:25 1997
Module Variables for each realization
Module Variables are stored for each realization
  13
WPCorrosionFailTime[yr]
Magn of Peak Annual Dose [rem/yr]
Time of Peak Annual Dose [yr]
Magn of Peak Annual Dose from Am243 [rem/yr]
....etc.....
  1      13
  0.21977E+04
  0.26855E-05
  0.10000E+05
  0.00000E+00
  ....etc.....
```

The first five lines are consistent with the general format described previously. Next, the number of module variables is specified. In this example, the number of variables is 13. The names of the module variables follow, one per line. These names are the same as introduced by the consequence modules. The next line has two integer values for the realization number and the number of variables (the number of module variable remain the same during a run). The specific values for the variables follows, one per line.

## 6.5 EPA SUMMED NORMALIZED CUMULATIVE RELEASE

The summed normalized cumulative release of radionuclides at the designated compliance boundary is calculated for the run and written to the file *epaccdf.tpa*. This information is normally plotted as a complementary cumulative distribution function (CCDF), hence the name of the file. The first five lines of the file are consistent with the general format described previously. On each subsequent line, the realization number and the summed normalized cumulative release are output, as shown below:

```
This is the test file for TPA 3.0beta version
which is being supplied to NRC on February 26, 1997.
TPA 3.0, Job started: Fri Mar 7 07:00:48 1997
Summed Normalized Release over 10,000 years
This is remanded EPA 40 CFR Part 191 Standard
  1      0.63141E+01
  ... etc ...
```

## 6.6 TIME-DEPENDENT ANNUAL EFFECTIVE DOSE EQUIVALENT

A number of files output time-dependent annual effective dose equivalent at the compliance point, including:

```
rgwnr.tpa
rgwsr.tpa
rgwna.tpa
rgwsa.tpa
rgsnr.tpa
rgssr.tpa
rgsna.tpa
rgssa.tpa
rgwgssa.tpa
```

These files are briefly described in table 6-1 which is discussed in the beginning of this section. The files break down as shown in figure 6-1. The files represent the pathway (either groundwater, ground surface, or both), the realization (either per realization or averaged over all realizations), and the nuclide (either per nuclide or summed over all nuclides).

The format of the files varies because of the need to report data per realization (or averaged over all realizations) and data per nuclide (or summed over all nuclides). The first few lines of the *rgwsa.tpa* file are shown below as an example:

```
This is the test file for TPA 3.0 BETA version
which is being supplied to NRC on January 31, 1997.
TPA 3.0, Job started: Tue Feb 25 17:19:44 1997
AEDE[rem/yr], GroundWater Pathway
summed over all nuclides, averaged over all realizations
  0.231E+01  0.100E-14
  0.467E+01  0.100E-14
  0.709E+01  0.100E-14
  0.957E+01  0.100E-14
... etc ...
```

The data have two columns, representing the time and the annual dose. The first few lines of this example file show no annual dose (the value has been set to a very small value of  $10^{-15}$  so this data can be plotted using a log scale).

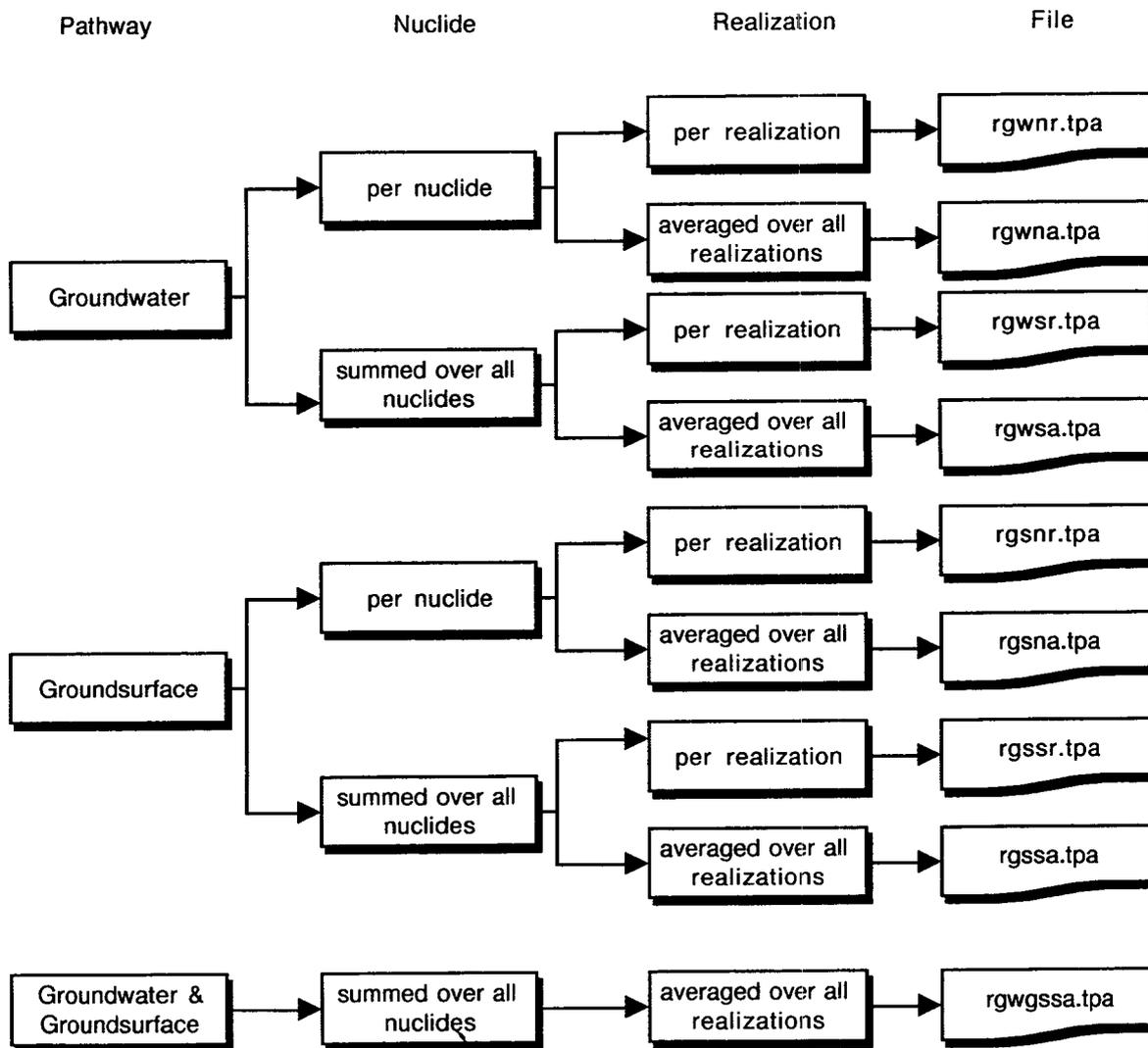


Figure 6-1. Summary of output files for annual effective dose equivalent

## 7 PROGRAM INSTALLATION AND EXECUTION

The TPA Version 3.0 code was developed on a Sun Microsystems, Inc. SPARC 20 Workstation. This platform uses a Sun Microsystem UNIX Operating System. This section describes the basic steps in the installation and execution of the TPA code.

### 7.1 INSTALLING SOURCE CODE FROM AN 8-MM TAPE

Because of the size of the source code, the TPA Version 3.0 code is generally supplied for installation on an 8-mm data cartridge tape that has been created using the *tar* command in the UNIX operating system environment. A 8-mm tape is a standard medium for archiving or transferring electronic data, especially between UNIX-based workstations. The FORTRAN source code, associated data files, and stand-alone programs have been written to the tape. The following command (possibly with some small variation) has been used to prepare the tape:

```
% tar cvf /dev/rst0 tpa
```

This command copies the *tpa* directory with all its files and subdirectories onto the tape. At the site where the program is to be installed, the user will type the following command (possibly with some small variation that depends on the specific system) to install the code:

```
% tar xvpf /dev/rmt/1n
```

This command unloads the tape and creates the *tpa* directory with all the necessary files, subdirectories, and programs starting at the location where the command is executed.

### 7.2 CUSTOMIZING THE CODE FOR THE USER'S UNIX OPERATING SYSTEM

If the user first installs the TPA Version 3.0 code or moves the source code and data files to another computer, then the user must modify the file *path.i*, which points to the archived source codes, data files, and stand-alone programs. Currently, the *path.i* file is as shown:

```
character*17 path  
data path / '/u01/tpaarch/tpa/' /
```

The *path.i* file may need to be changed to something such as the following:

```
character*25 path  
data path / '/datax/home/nmss2/ymipa3/' /
```

If the path is changed, all the object files should be deleted, and the Makefile executed to generate a new TPA Version 3.0 code executable. This procedure is described in the next section.

### 7.3 PROGRAM EXECUTION

The TPA Version 3.0 code is written in FORTRAN 77 and designed to run under the Sun Microsystems, Inc. UNIX operating system. The UNIX operating system supports many features to assist

code developers. One feature is the Makefile, which allows the developer to automatically link the objects of many subroutines that reside in more than one file. An example of a Makefile is shown below (and also provided on the 8-mm tape):

```
OBJECTS = array.o \
          condxyzt.o \
          dcagw.o \
          dcags.o \
          ebsfailg.o \
          ebsfail.o \
          ebsrel.o \
          faulto.o \
          fileunit.o \
          findelev.o \
          invent.o \
          mv.o \
          nfenv.o \
          numrecip.o \
          ran.o \
          reader.o \
          sampler.o \
          seismo.o \
          subarea.o \
          szft.o \
          uzflow.o \
          uzft.o \
          volcano.o

tpa : $(OBJECTS)
      f77 exec.f -o tpa.e $(OBJECTS)
```

After unloading the TPA Version 3.0 code from the tape and modifying the *path.i* file, the user can generate an executable version of the code by typing:

```
% make tpa
```

This command will prompt the UNIX system to compile the *exec.f* file and link with the 23 listed object files. The Makefile also has the handy feature of ensuring the object files are up to date. The Makefile will check the time and date of the last change made to the 23 FORTRAN source code files used to create the object files. If any of the FORTRAN source code files have been changed since the object files were made, then the object files are considered out of date. The Makefile initiates a compilation of the FORTRAN source code to generate a new object file. This procedure is performed on a file-by-file basis, making it an efficient process. Only the files that have been changed are compiled. The Makefile then generates an executable file named *tpa.e*. To execute the code, the user then would type:

```
% tpa.e
```

The TPA Version 3.0 code will open and read the file *tpa.inp*, so it must reside where the user is executing *tpa.e*. The TPA Version 3.0 code requires only the dynamic input data file. The TPA code will generate a number of output files that all end with *.tpa*. These files are discussed in chapter 6 of this report.

Some consequence modules spawn jobs that execute stand-alone programs (e.g., NEFTRAN). The source code for these stand-alone programs is provided in the subdirectory standalonedecodes. The user may need to compile these programs using a command such as:

```
% f77 -r8 -C nefmks97.f -o nefmks.e
```

The user is cautioned not to make changes to the stand-alone source code files, because these are part of the TPA Version 3.0 code. If changes are needed, these need to be coordinated with the software QA code custodian.

## 7.4 PORTABILITY LIMITATION OF TPA VERSION 3.0

The TPA code is written in FORTRAN 77 as implemented in the Sun compiler version 4.2 (SunSoft, 1996). The subroutines have been designed and written to be as portable as possible, recognizing that some nonstandard FORTRAN features are supported on many computer platforms, while some features are not supported on a wide variety of platforms. For example, the time and date functions are nonstandard FORTRAN 77, but are required for QA documentation purposes. Fortunately, many FORTRAN compilers on a wide variety of computer platforms have the built-in nonstandard feature of time and date. Similarly, many compilers support the use of variable and array names longer than six characters, so in the TPA code, long variable names are used to help add clarity to the source code.

One nonstandard feature, considered essential by consequence module developers is the spawning (i.e., running) of external stand-alone programs such as NEFTRAN. To implement this program, the operating system must support the capability to spawn external jobs. This feature is not widely supported among computer system, especially on the IBM PC platforms. At this time, the need to spawn external programs prevents the TPA Version 3.0 code from being easily converted to run on platforms using Windows or DOS Operating Systems (this includes most personal computers). This portability limitation also applied to the TPA Version 2.0 code, because it also spawned jobs to non external programs.

## 7.5 USER SUPPORT

For technical assistance, users may contact:

S. Mohanty  
Center for Nuclear Waste Regulatory Analyses  
Southwest Research Institute  
P.O. Drawer 28510  
San Antonio, TX 78228-0510  
(210) 522-5185

or

R.G. Baca  
Center for Nuclear Waste Regulatory Analyses  
Southwest Research Institute  
P.O. Drawer 28510  
San Antonio, TX 78228-0510  
(210) 522-3805

## 7.6 SOFTWARE QUALITY ASSURANCE

Software QA has been an integral part of the TPA Version 3.0 code development process. Many of the software QA recommendations made in Wescott et al. (1995) were implemented in developing the TPA Version 3.0 code. This testing process was conducted in an independent manner by staff that did not contribute to the module development. Module testing was performed not only to probe for the correct implementation and determine the limitations of computational algorithms, but also to assess code capabilities to detect user input errors. The executive, consequence, and utility modules were put through a broad spectrum of tests considered appropriate for the specific module. Specific types of tests performed included:

- Verification of utility modules through comparisons with calculations made with MCAD (Mathsoft Inc., 1994) and Mathematica (Wolfram, 1991)
- Verification of module outputs through comparison of results from the stand-alone module and the same module integrated into TPA Version 3.0 code
- Verification of selected modules through hand calculations and published results, where deemed feasible
- Line-by-line checking of libraries containing static data for parameters such as radionuclide half-lives, specific activities, inventories, and EPA limits
- Line-by-line checking of selected consequence modules (e.g., NFENV), where deemed feasible
- Compilation of module source code using different FORTRAN compilers

In addition, full runs were made with the TPA Version 3.0 code and the results (i.e., subsystem and total-system performance measures) checked for reasonableness.

The major part of the module testings has been captured in notebooks for the software QA record. In addition, the 8-mm tape provided with the source code contains a directory with: (i) README text files that contain summaries of the module testing, (ii) source code for the drivers and input files used to test the modules, and (iii) output files of the test results. These are provided in the event that additional testing is warranted to check module modifications made in the future.

Because of the size and complexity of the TPA Version 3.0 code, further testing is planned and will be conducted prior to use of the code in reviewing the DOE TSPA-VA and license application. The TPA Version 3.0 code is currently being placed under software configuration in accordance with the CNWRA Technical Operating Procedure, TOP-018.

## 8 REFERENCES

- Ahn, T., and P. Soo. 1983. *Container Assessment—Corrosion Study of HLW Container Materials, Quarterly Progress Report, October–December 1983*. BNL-NUREG-34220. Upton, NY: Brookhaven National Laboratory.
- Ahn, T., and P. Soo. 1984. *Container Assessment—Corrosion Study of HLW Container Materials, Quarterly Progress Report, January–March 1984, April–June 1984, July–September 1984*. (Informal Reports. Limited Distribution.) Upton, NY: Brookhaven National Laboratory.
- Ahola, M.P., R.B. Hoffman, S.M. Hsiung, and R.D. Manteufel. 1995. Numerical Study of Seismic Motion Effects on Drift Stability. *Focus '95: Methods of Seismic Hazard Evaluation and Seismic Design*. LaGrange Park, IL: American Nuclear Society: 184–194.
- Amtec Engineering, Inc. 1993. *TECPLOT Interactive Data Visualization for Scientists and Engineers, Version 6 Users Manual*. Bellevue, WA: Amtec Engineering Inc.
- 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 and Sons.
- Anspaugh, L.R., Shinn, J.H., Phelps, P.L., and N.C. Kennedy. 1975. Resuspension and redistribution of plutonium in soils. *Health Physics* 29: 571–582.
- Baca, R.G., G.W. Wittmeyer, and R.W. Rice. 1996. Analysis of Contaminant Dilution in Groundwater. *Scoping Calculations for Revisions to the EPA Standard*. NUREG-1538. Washington, DC: Nuclear Regulatory Commission.
- Baes, C.F., III, R.D. Sharp, A.L. Sjoreen, and R.W. Shor. 1982. *A Review and Analysis of Parameters for Assessing Transport of Environmentally Released Radionuclides Through Agriculture*. ORNL-5786. Oak Ridge, TN: Oak Ridge National Laboratories.
- Barnard, R.W., M.L. Wilson, H.A. Dockery, J.W. Gauthier, P.G. Kaplan, R.R. Easton, F.W. Bingham, and T.H. Robey. 1992. *TSPA 1991: An Initial Total-System Performance Assessment for Yucca Mountain*. SAND 91-2795. Albuquerque, NM: Sandia National Laboratories.
- Bateman, H. 1910. The solution of a system of differential equations occurring in the theory of radioactive transformation. *Proceedings of the Cambridge Philosophical Society* 15: 473.
- Benjamin, J.R., and C.A. Cornell. 1970. *Probability, Statistics, and Decision for Civil Engineers*. New York: McGraw-Hill.
- Bonano, E.J., and R.G. Baca. 1994. *Review of Scenario Selection Approaches for Performance Assessment of High-Level Waste Repositories and Related Issues*. CNWRA 94-002. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.
- Boyce, W.E., and R.C. DiPrima. 1977. *Elementary Differential Equations and Boundary Value Problems*. New York: Wiley.

- Breshears, D.D., T.B. Kirchner, M.D. Otis, and F.W. Whicker. 1989. Uncertainty in predictions of fallout radionuclides in foods and of subsequent ingestion. *Health Physics* 57(6): 943-953.
- Briant, C.L., and S.K. Banerji. 1983. Intergranular fracture in ferrous alloys in nonaggressive environments. *Treatise on Materials Science and Technology. Volume 25: Embrittlement of Engineering Alloys*. C.L. Briant and S.K. Banerji, eds. New York, NY: Academic Press: 21-58.
- Carslaw, H.S., and J.C. Jaeger. 1959. *Conduction of Heat in Solids*. London: Oxford University Press.
- Cember, H. 1983. *Introduction to Health Physics*. Pergamon Press: New York, NY.
- Claesson, J., and T. Proberts. 1996. *Temperature field due to time-dependent heat sources in a large rectangular grid—Derivation of analytical solution*. SKB 96-12. Stockholm, Sweden: Swedish Nuclear Fuel and Waste Management Co.
- Codell, R.B., N.A. Eisenberg, D.J. Fehring, W.H. Ford, T.S. Margulies, T.J. McCartin, J.R. Park, and J.D. Randall. 1992. *Initial Demonstration of the NRC's Capability to Conduct a Performance Assessment for a High-Level Waste Repository*. NUREG-1327. Washington, DC: Nuclear Regulatory Commission.
- Condit, C.D., and C.B. Connor. 1996. Recurrence rate of basaltic volcanism in volcanic fields: An example from the Springerville Volcanic Field, Arizona. *Geological Society of America, Bulletin* 108: 1,225-1,241.
- Connor, C.B., L. Powell, W. Strauch, M. Navarro, O. Urbina, and W.I. Rose. 1993. *The 1992 eruption of Cerro Negro, Nicaragua: An example of Plinian-style activity at a small basaltic cinder cone*. EOS, *Transactions of the American Geophysical Union* 74(43): 640.
- Connor, C.B., and B.E. Hill. 1993. Estimating the probability of volcanic disruption at the Yucca Mountain site using nonhomogeneous Poisson models. *Focus '93 LaGrange Park, IL: American Nuclear Society*: 174-181.
- Connor, C.B., and B.E. Hill. 1995. Three nonhomogeneous Poisson models for the probability of basaltic volcanism: Application to the Yucca Mountain region. *Journal of Geophysical Research* 100: 10,107-10,125.
- Connor, C.B., J.A. Stamatakos, D.A. Ferrill, B.E. Hill, S.L. Magsino, P. LaFemina, and R.H. Martin. 1996. Integrating structural models into volcanic hazard assessments: an example from Yucca Mountain, Nevada. EOS *Transactions of the American Geophysical Union* 77: F20.
- Connor, C.B., S. Magsino, J. Stamatakos, R. Martin, P. La Femina, B. Hill, and S. Lieber. 1997. Magnetic surveys help reassess volcanic hazards at Yucca Mountain. EOS, *Transactions of the American Geophysical Union* 74(7): 73-78.

- Conway, M.F., D.A. Ferrill, C.M. Hall, A.P. Morris, J.A. Stamatakos, C.B. Connor, A.N. Halliday, and C. Condit. 1997. Timing of basaltic volcanism along the Mesa Butte fault zone in the San Francisco volcanic field, Arizona, from  $^{40}\text{Ar}/^{39}\text{Ar}$  ages: Implications for longevity of cinder cone alignments. *Journal of Geophysical Research* 102: 815–824.
- Cragolino, G.A., N. Sridhar, J. Walton, R. Janetzke, T. Torng, J. Wu, and P. Nair. 1994. *Substantially Complete Containment—Example Analysis of a Reference Container*. CNWRA 94-003. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.
- Cranwell R.M., R.V. Guzowski, J.E. Campbell, and N.R. Ortiz. 1990. *Risk Methodology for Geological Disposal of Radioactive Waste: Scenario Selection Procedure*. NUREG/CR-1667. Washington, DC: Nuclear Regulatory Commission.
- Croff, A.G. 1983. ORIGEN2: A Versatile Computer Code for Calculating Nuclide Compositions and Characteristics of Nuclear Materials. *Nuclear Technology* 62: 335–352.
- Crowe, B.M., M.E. Johnson, and R.J. Beckman. 1982. Calculation of the probability of volcanic disruption of a high-level nuclear waste repository within southern Nevada, USA. *Radioactive Waste Management and the Nuclear Fuel Cycle* 3: 167–190.
- Crowe, B.M., R. Picard, G.A. Valentine, and R.F. Perry. 1992. Recurrence models of volcanic events: Applications to volcanic risk assessment. *Proceedings of the Third International Conference on High-Level Radioactive Waste Management*. LaGrange Park, IL: American Nuclear Society: 2,344–2,355.
- Czarnecki, J.B., and R.K. Waddell. 1984. *Finite-Element Simulation of Groundwater Flow in the Vicinity of Yucca Mountain, Nevada-California*. Water Resources Investigations Report WRI-84-4349. Denver, CO: U.S. Geological Survey.
- Daily, M.C., A. Huffert, F. Cardile, and J.C. Malaro. 1994. *Working Draft Regulatory Guide on Release Criteria for Decommissioning: NRC Staff's Draft for Comment*. NUREG-1500. Washington, DC: Nuclear Regulatory Commission.
- Davis, P.A., L.L. Price, K.K. Wahi, M.T. Goodrich, D.P. Gallegos, E.J. Bonano, R.V. Guzowski. 1990. *Components of an Overall Performance Assessment Methodology*. NUREG/CR-5256, SAND88-3020. Albuquerque, NM: Sandia National Laboratories.
- Delaney, P.T., and A.E. Gartner. 1995. *Physical Processes of Shallow Mafic Dike Emplacement near the San Rafael Swell, Utah*. USGS Open-File Report 95-491. Reston, VA: U.S. Geological Survey.
- DeWispelare, A.R., L.T. Herren, M.P. Miklas, and R.T. Clemen. 1993. *Expert Elicitation of Future Climate in the Yucca Mountain Vicinity*. CNWRA 93-016. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.
- Dunn, D.S., G.A. Cragolino, H.K. Manaktala, P. Angell, Y.-M. Pang, and N. Sridhar. 1996. Factors influencing the performance of carbon steel overpacks in the proposed high-level nuclear waste repository. *CORROSION* 97. Houston, TX: NACE International: Paper No. 99. To be published.

- Einzigler, R.E., L.E. Thomas, H.C. Buchanan, and R.B. Stout. 1992. Oxidation of spent fuel in air at 175 to 195 °C. *Journal of Nuclear Materials* 190: 53–60.
- England, R.L., K.J. Ekblad, and R.G. Baca. 1985. *MAGNUM-2D Computer Code: User's Guide*. RHO-BW-CR-143. Richland, WA: Rockwell International.
- Fyfe, D. 1994. *Corrosion*. R.A. Jarman, and G.T. Burstein, eds. Oxford, UK: Butterworth Heinmann: 1: 2:31–2:42
- Geomatrix, 1996. *Probabilistic Volcanic Hazard Analysis of Yucca Mountain, Nevada*. Geomatrix Consultants, San Francisco, CA: Rep. BA0000000-1717-2200-00082.
- Ghosh, A., R.D. Manteufel, and G.L. Stirewalt. 1997. *FAULTING Version 1.0—A Code for Simulation of Direct Fault Disruption: Technical Description and User's Guide*. CNWRA 97-002. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.
- Glaze L.S., and S. Self. 1991. Ashfall Dispersal for the 16 September 1986 Eruption of Lascar, Chile, Calculated by a Turbulent Diffusion Model. *Geophysical Research Letters* 18: 1,237–1,240.
- Heffter, J.L., and B.J.B. Stunder. 1993. Volcanic Ash Forecast Transport and Dispersion (VAFTAD) Model. *Weather and Forecasting* 8: 533–541.
- Hill, B.E. 1996. *Constraints on the Potential Subsurface Area of Disruption Associated with Yucca Mountain Region Basaltic Volcanoes*. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.
- Hill, B.E., and C.B. Connor. 1996. Volcanic Systems of the Basin and Range. *NRC High-Level Radioactive Waste Research at CNWRA, July–December 1995*. CNWRA 95–02S. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses: 5-1 to 5-21.
- Hill, B.E., C.B. Connor, and J.S. Trapp. 1996. Igneous activity. *NRC High-Level Radioactive Waste Program Annual Progress Report, Fiscal Year 1996*. CNWRA 96-01A. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses. 2-1 to 2-32.
- Ho, C.-H. 1991. Time trend analysis of basaltic volcanism at the Yucca Mountain site. *Journal of Volcanology and Geothermal Research* 46: 61–72.
- Ho, C.H., E.I. Smith, D.L. Feuerbach, and T.R. Naumann. 1991. Eruptive probability calculation for the Yucca Mountain site, USA: statistical estimation of recurrence rates. *Bulletin Volcanology* 54: 50–56.
- Hoffman, F.O., Gardner, R.H., and K.F. Eckerman. 1982. *Variability in Dose Estimates Associated with the Food Chain Transport and Ingestion of Selected Radionuclides*. NUREG/CR-2612. Washington, DC: Nuclear Regulatory Commission.
- Hopkins, A.T., and C.J. Bridgeman. 1985. A volcanic ash transport model and analysis of Mount St. Helens ashfall. *Journal of Geophysical Research*. 90: 10,620–10,630.

- 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*. NUREG/CR-3624 Washington, DC: Nuclear Regulatory Commission.
- Incropera, F.P., and D.P. DeWitt, 1990. *Fundamentals of Heat and Mass Transfer*. Third Edition. New York: John Wiley & Sons.
- International Atomic Energy Agency. 1994. *Handbook of Parameter Values for the Prediction of Radionuclide Transfer in Temperate Environments*. Technical Report Series No. 364. Vienna, Austria: International Atomic Energy Agency.
- Itasca Consulting Group, Inc. 1996. *UDEC—Universal Distinct Element Code, Version 3, Volume I: User's Manual*. Minneapolis, MN: Itasca Consulting Group, Inc.
- Jarzemba, M.S. 1996. Stochastic radionuclide distributions after a basaltic eruption for performance assessments of Yucca Mountain. *Nuclear Technology*. In press.
- Jarzemba, M.S., and P.A. LaPlante. 1996. *Preliminary Calculations of Expected Dose from Extrusive Volcanic Events at Yucca Mountain*. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.
- Jarzemba, M.S., P.A. LaPlante, and K.J. Poor. 1997. *Ashplume Code Version 1.0 Model Description and User's Guide*. CNWRA 97-004. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.
- Kennedy, W.E., and D.L. Strenge. 1992. *Residual Radioactive Contamination From Decommissioning: Technical Basis for Translating Contamination Levels to Annual Total Effective Dose Equivalent*. NUREG/CR-5512. Vol. 1. Washington, DC: Nuclear Regulatory Commission.
- LaPlante, P.A., S.J. Maheras, and M.S. Jarzemba. 1995. *Initial Analysis of Selected Site-Specific Dose Assessment Parameters and Exposure Pathways Applicable to a Groundwater Release Scenario at Yucca Mountain*. CNWRA 95-018. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.
- Leigh, C.D., Thompson, B.M., Campbell, J.E., Longsine, D.E., Kennedy, R.A., and B.A. Napier. 1993. *User's Guide for GENII-S: A Code for Statistical and Deterministic Simulation of Radiation Doses to Humans from Radionuclides in the Environment*. SAND 91-0561. Albuquerque, NM: Sandia National Laboratories.
- Leygraf, C. 1995. Atmospheric Corrosion. *Corrosion Mechanisms in Theory and Practice*. P. Marcus and J. Oudar, eds. New York: Marcel Dekker, Inc.: 421–455.
- Lichtner, P.C., and M.S. Seth. 1996. *User's Manual for MULTIFLO: Part II—MULTIFLO 1.0 and GEM 1.0. Multicomponent-Multiphase Reactive Transport Model*. CNWRA 96-010. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.

- Liskov, B., and J. Guttag. 1986. *Abstraction and Specification in Program Development*. Cambridge, MA: MIT Press.
- Lobnig, R.E., H.P. Schmidt, K. Hennesen, and H.J. Grabke. 1992. Diffusion of cations in Chromia layers grown on iron-base alloys. *Oxidation of Metals* 37: 81-93.
- Lozano, A.S., H. Karimi, J.P. Cornelius, R.D. Manteufel, and R.W. Janetzke. 1994. *INVENT: A Module for the Calculation of Radionuclide Inventories, Software Description, and User Guide*. CNWRA 94-016. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.
- Ludwig, S.B., and J.P. Renier. 1989. *Standard and Extended-Burnup PWR and BWR Reactor Models for the ORIGEN2 Computer Code*. ORNL/TM-11018. Oak Ridge, TN: Oak Ridge National Laboratory.
- Lutz, T.M., and J.T. Gutmann. 1995. An improved method of determining alignments of point-like features and its implications for the Pinacate volcanic field, Mexico. *Journal of Geophysical Research* 100: 17,659-17,670.
- Manteufel, R.D., and N.E. Todreas. 1994. Effective thermal conductivity and edge conductance model for a spent fuel assembly. *Nuclear Technology* 105(3): 421-440.
- Manteufel, R.D. 1997. Effects on ventilation and backfill on a mined waste disposal facility. *Nuclear Engineering and Design*. Accepted for publication.
- Manteufel, R.D., R.G. Baca, M.P. Ahola, A.C. Bagtzoglou, E.J. Bonano, B. Gureghian, R.B. Hofmann, S. Hsiung, R.W. Janetzke, M.S. Jarzempa, K. Karimi, P.A. LaPlante, P.C. Lichtner, W.M. Murphy, D.A. Pickett, G.L. Stirewalt, and S.A. Stathoff. 1995. *Iterative Performance Assessment Phase 3: Status of Activities*. CNWRA 95-007. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.
- Margulies, T., L. Lancaster, N. Eisenberg, and L. Abramson. 1992. Probabilistic analysis of magma scenarios for assessing geologic waste repository performance. *American Society of Mechanical Engineers, Winter Annual Meeting*, November 8-13, 1992, Anaheim, CA.
- MathSoft Inc., 1995. *MathCad Plus 5.0 Users Guide*. Cambridge, MA: MathSoft Inc.
- Microsoft Corporation. 1997. *Fortran Development System for MS-DOS and Windows, Version 5.1*. Redmond WA: Microsoft Corporation.
- Mohanty S., G.A. Cragolino, T. Ahn, D.S. Dunn, P.C. Lichtner, R.D. Manteufel, and N. Sridhar. 1996. *Engineered Barrier System Performance Assessment Code: EBSPAC Version 1.0 BETA, Technical Description and User's Manual*. CNWRA96-001. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.

- Napier, B.A., R.A. Peloquin, D.L. Strenge, and J.V. Ramsdell. 1988. *GENII: The Hanford Environmental Radiation Dosimetry Software System, Volumes 1, 2, and 3: Conceptual Representation, User's Manual, Code Maintenance Manual*. PNL-6584, Volumes 1, 2, and 3. Richland, WA: Pacific Northwest Laboratory.
- National Academy of Sciences. 1995. *Technical Bases for Yucca Mountain Standards*. Washington, DC: National Academy Press.
- Nevada Agricultural Statistics Service. 1988. *Nevada Agricultural Statistics: 1, 1987-88*. Reno, NV: Nevada Agricultural Statistics Service.
- Nevada Division of Water Resources. 1995. *Preliminary Special Hydrographic Abstract for Valley Basin No. 230 from the Nevada Division of Water Resources Water Rights Database*. Carson City, NV: Nevada Division of Water Resources. Unpublished.
- Nuclear Regulatory Commission. 1977. *Regulatory Guide 1.109: Calculation of Annual Doses to Man From Routine Releases of Reactor Effluents for the Purpose of Evaluating Compliance with 10 CFR Part 50, Appendix I*. Revision 1. Washington DC: Nuclear Regulatory Commission.
- Nuclear Regulatory Commission. 1994. *Policy and Guidance Directive PG-8-08: Scenarios for Assessing Potential Doses Associated with Residual Radioactivity*. PG-8-08. Washington, DC: Nuclear Regulatory Commission.
- Office of the Federal Register. 1989. *Environmental Radiation Protection Standards for Management and Disposal of Spent Nuclear Fuel, High-Level and Transuranic Radioactive Wastes*. Title 40—Energy, Chapter 1—Nuclear Regulatory Commission, Part 191. Washington, DC: U.S. Government Printing Office.
- Office of the Federal Register. 1992. *Disposal of High-Level Radioactive Wastes in Geologic Repositories*. Title 10—Energy, Chapter 1—Nuclear Regulatory Commission, Part 60. Washington, DC: U.S. Government Printing Office.
- Oishi, Y., and H. Ichimura. 1979. Grain-boundary enhanced interdiffusion in polycrystalline CaO-stabilized Zirconia system. *Journal of Chemical Physics* 71(12): 5,134–5,139.
- Olague, N.E., D.E. Longsine, J.E. Campbell, and C.D. Leigh. 1991. *User's Manual for the NEFTRAN II Computer Code*. NUREG/CR-5618. Washington, DC: Nuclear Regulatory Commission.
- Otis, M.D. 1983. *Sensitivity and Uncertainty Analysis of the PATHWAY Radionuclide Transport Model*. Ph.D. Dissertation. Fort Collins, CO: Colorado State University.
- Popov, E.P. 1970. *Mechanics of Materials*. New York: Prentice Hall.
- Press, W.H., B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. 1986. *Numerical Recipes in FORTRAN*. Cambridge, UK: Cambridge University Press.
- Ripley, B.D. 1987. *Stochastic Simulation*. New York, NY: John Wiley and Sons.

- Rolfe, S.T., and J. M. Barson. 1977. *Fracture and Fatigue Control in Structures*. Prentice-Hall. Englewood Cliffs, NJ: 156.
- Roxburgh, I.S. 1987. *Geology of High-Level Nuclear Waste Disposal, An Introduction*. New York: Chapman and Hall.
- Sagar, B., R.B. Codell, J. Walton, and R.W. Janetzke. 1992. *SOTEC: A Source Term Code for High-Level Nuclear Waste Geologic Repositories User's Manual: Version 1.0*. CNWRA 92-009. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.
- Sagar, B., and R.W. Janetzke. 1993. *Total-System Performance Assessment (TPA) Computer Code: Description of Executive Module, Version 2.0*. CNWRA 93-017. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.
- Scott, R.B. 1990. *Tectonic setting of Yucca Mountain, southwest Nevada. Basin and Range Extensional Tectonics Near the Latitude of Las Vegas, Nevada*. B.P. Wernicke, ed. Geological Society of America Memoir 176. 251-282.
- Scott, R.B., and J. Bonk. 1984. *Preliminary Geologic Map (1:12,000 scale) of Yucca Mountain, Nye County, Nevada, with Geologic Cross Sections*. U.S. Geological Survey Open-File Report 84-494. Denver, CO: U.S. Geological Survey.
- Scully, J.R., and H.P. Hack. 1984. Galvanic corrosion prediction using long- and short-term polarization curves. *CORROSION 84*. Paper No. 34. Houston, TX: National Association of Corrosion Engineers.
- Sheridan, M.F. 1992. A Monte Carlo technique to estimate the probability of volcanic dikes. *High-Level Radioactive Waste Management: Proceedings of the Third International Conference*. LaGrange Park, IL: American Nuclear Society 2: 2,033-2,038.
- Smith, E.I., T.R. Naumann, D.L. Feuerbach, and J.E. Faulds. 1990. The area of most recent volcanism near Yucca Mountain, Nevada: implications for volcanic risk assessment. *High-Level Radioactive Waste Management: Proceedings of the First International Conference*. Las Vegas, NV. American Nuclear Society: 81-90.
- Snyder, S.F., T.A. Ikenberry, W.T. Farris, R.O. Gilbert, and B.A. Napier. 1994. *Parameters Used in the Environmental Pathways and Radiological Dose Modules (DESCARTES, CIDER, and CRD Codes) of the Hanford Environmental Dose Reconstruction Integrated Code (HEDRIC)*. PNWD-2023 HEDR. Rev. 1. Richland, WA: Pacific Northwest Laboratory.
- Spengler, R.W., C.A. Braun, L.G. Martin, and C.W. Weisenberg. 1994. *The Sundance Fault—A Newly Recognized Shear Zone at Yucca Mountain, Nevada*. U.S. Geological Survey Open-File Report 94-49. Denver, CO: U.S. Geological Survey.
- Sridhar, N., G.A. Cragnolino, and D.S. Dunn. 1993. *Experimental Investigations of Localized Corrosion of High-Level Waste Container Materials*. CNWRA 93-004. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.

- Stothoff, S.A., H.M. Castellaw, and A.C. Bagtzoglou. 1997. Simulating the Spatial Distribution of Infiltration at Yucca Mountain, Nevada. *Water Resources Research*. Submitted for publication.
- SunSoft, Inc. 1996. *Fortran 77 Language Reference, Compiler 4.2*. Mountain View, CA: SunSoft, Inc.
- Suzuki, T. 1983. *A theoretical model for dispersion of tephra*. *Arc Volcanism: Physics and Tectonics*. Tokyo, Japan. Terra Scientific Publishing: 95–113.
- Swadley, W.C., D.L. Hoover, and M.N. Rosholt. 1984. *Preliminary Report on Late Cenozoic Faulting and Stratigraphy in the Vicinity of Yucca Mountain, Nye County, Nevada*. U.S. Geological Survey Open-File Report 84-788. Denver, CO: U.S. Geological Survey.
- Thompson, B.G.J. 1988. *A Method for Overcoming the Limitation of Conventional Scenario-Based Assessments by Using Monte Carlo Simulation of Possible Future Environmental Changes*. PAAG/DOC/88/11. Paris, France: Nuclear Energy Agency/Organization for Economic Cooperation and Development.
- Thompson, B.G.J., and B. Sagar. 1993. The development and application of integrated procedures of post-closure assessment, based upon Monte Carlo simulation: the probabilistic systems assessment (PSA) approach. *Reliability Engineering and System Safety* 42: 125–160.
- Timoshenko, S.P., and J.N. Goodier. 1987. *Theory of Elasticity*. McGraw-Hill, New York.
- Tokarev, P.I. 1983. Calculation of the magma discharge, growth in the height of the cone and dimensions of the feeder channel of Crater I in the Great Tolbachik Fissure Eruption, July 1975. *The Great Tolbachik Fissure Eruption, Geological and Geophysical data, 1975-1976*. S.A. Fedotov and Ye. K. Markhnin, eds. Cambridge. Cambridge University Press: 27–35.
- TRW Environmental Safety Systems Inc. 1995. *Total System Performance Assessment — 1995: An Evaluation of the Potential Yucca Mountain Repository*. B00000000-01717-2200-00136, Rev. 01. Las Vegas, NV: TRW Environmental Safety Systems Inc.
- TRW Environmental Safety Systems Inc. 1996. *Total System Performance Assessment—Viability Assessment (TSPA-VA)*. B00000000-01717-2200-00179. Las Vegas, NV: TRW Environmental Safety Systems Inc.
- U.S. Department of Commerce. 1989. *Census of Agriculture, Volume 1: Geographic Area Series, Part 28: Nevada State and County Data*. AC87-A-28. Washington DC: U.S. Department of Commerce.
- U.S. Department of Energy. 1988. *Site Characterization Plan: Yucca Mountain Site, Nevada Research and Development Area, Nevada*. DOE/RW-0199. Washington, DC: Office of Civilian Radioactive Waste Management, Department of Energy.
- U.S. Department of Energy. 1996. *Mined Geologic Disposal System Advance Conceptual Design Report*. B00000000-01717-5705-00027 Rev 00. Washington, DC: Office of Civilian Radioactive Waste Management. Department of Energy.

- U.S. Environmental Protection Agency. 1988. *Limiting Values of Radionuclides Intake and Air Concentration and Dose Conversion Factors for Inhalation, Submersion, and Ingestion*. EPA-520/1-88-020. Washington, DC: U.S. Environmental Protection Agency.
- van Genuchten, M.T. 1980. A Closed-Form Equation for Predicting the Hydraulic Conductivity of Unsaturated Soils. *Soil Science Society of America Journal* 44: 892-898.
- Vander Voort, G.F. 1990. Embrittlement of steels. *Metals Handbook, Vol. 1, 10th edition, Properties and Selection: Irons, Steels, and High-Performance Alloys*. Materials Park, OH: ASM International: 689-736.
- Wescott, R.G., M.P. Lee, T.J. McCartin, N.A. Eisenberg, and R.G. Baca. 1995. *NRC Iterative Performance Assessment Phase 2: Development of Capabilities for Review of a Performance Assessment for a High-Level Waste Repository*. NUREG-1464. Washington, DC: Nuclear Regulatory Commission.
- Wilson, M.L., J.H. Gauthier, R.W. Barnard, G.E. Barr, H.A. Dockery, E. Dunn, R.R. Eaton, D.C. Guerin, N. Lu, M.J. Martinez, R. Nilson, C.A. Rautman, T.H. Robey, B. Ross, E.E. Ryder, A.R. Schenker, S.A. Shannon, L.H. Skinner, W.G. Haley, J.D. Gansemer, L.C. Lewis, A.D. Lamont, I.R. Triay, A. Meiker, and D.E. Morris. 1994. *Total-System Performance Assessment for Yucca Mountain-SNL Second Iteration (TSPA-93)*. SAND 93-2675, Vol. 1 and 2. Albuquerque, NM: Sandia National Laboratories.
- Wolfram, S. 1991. *Mathematica: A System for Doing Mathematics by Computer*. Redwood City, CA: Addison-Wesley Publishing Company.
- Yogodzinski, G.M., and E.I. Smith. 1995. Isotopic domains and the area of interest for volcanic hazard assessment in the Yucca Mountain area. *EOS Transactions of the American Geophysical Union* 76(46): F669.
- Yokoyama, I., and S. De la Cruz-Reyna. 1990. Precursory earthquakes of the 1943 eruption of Parícutin volcano, Michoacán, Mexico. *Journal of Volcanology and Geothermal Research* 44: 265-281.

**APPENDIX A**

**ORIGINAL SOFTWARE  
REQUIREMENTS DESCRIPTION  
FOR TPA VERSION 3.0 CODE**

# **ORIGINAL SOFTWARE REQUIREMENTS DESCRIPTION FOR TPA VERSION 3.0 CODE**

## **1 INTRODUCTION**

This software requirements description is the first step in updating the Total Performance Assessment (TPA) code from version 2.0 to 3.0. The TPA code Version 2.0 (Sagar and Janetzke, 1993) was used in the Nuclear Regulatory Commission/Center for Nuclear Waste Regulatory Analyses (NRC/CNWRA) Iterative Performance Assessment (IPA) Phase 2 exercise. The TPA code is an executive module and a set of consequence modules that simulate the performance of a geologic repository of nuclear high-level waste (HLW) at Yucca Mountain (YM), Nevada. The executive module controls the flow of data and execution between the process/component-specific consequence modules that simulate major safety components of the repository system. The TPA code integrates geologic site characterization data, proposed repository and waste package (WP) engineered designs, and biosphere data. The consequence modules are designed with input from materials engineers, hydrogeologists, seismologists, volcanologists, rock mechanics, and health physicists. Recent developments in the proposed YM repository necessitate a new version of TPA code. These developments include (i) new repository, WP, and emplacement designs, (ii) changing regulatory standards (from release-based to dose-based), and (iii) potentially longer time periods of concern (to hundreds of thousands of years). Numerous improvements will be incorporated that reflect knowledge and data gained in recent years of site characterization and laboratory studies, as well as other total system performance assessments (TSPA). The TPA code will have the capability to assess the compliance of the proposed YM repository with regulatory requirements using a probabilistic approach to account for uncertainties.

## **2 SOFTWARE FUNCTION**

The TPA code is a combination of an executive module and a set of consequence modules that stochastically assess the overall performance of the proposed YM HLW repository with applicable regulatory standards. The executive driver controls the probabilistic sampling of input parameters, the calculational flow process between modules, and the generation of output files. Output files can be used for parameter importance analyses, generation of time-dependent risk curves, and generation of complementary cumulative distribution functions for cumulative release of radionuclides. Utility modules ensure a consistent description of the proposed repository system and flow of data between consequence modules. Examples include the spatial and temporal discretizations (i.e., subarea (SA) discretization of the proposed repository and time stepping scheme). In the NRC/CNWRA IPA Phase 2 exercise, there were 7 SAs and 50 time steps of 200 yr each. The number of SAs in the TPA code will be based on the latest proposed repository design and reflect near-field thermal-hydrologic-mechanical-chemical (THMC) environments in the proposed repository as well as hydrostratigraphy. The time stepping will be variable over the simulation as well as the total time period of interest (TPI).

## **3 TECHNICAL AND COMPUTATIONAL APPROACHES**

The overall conceptual approach of a TSPA is outlined in figure 1. Data flow from the system characterization to final regulatory compliance determination (shown as either a cumulative release or dose). The bulk of the modeling effort is in the consequence modules that include both anticipated processes (also called base-case processes) and disruptive processes. The base-case system has seven major subsystem models:

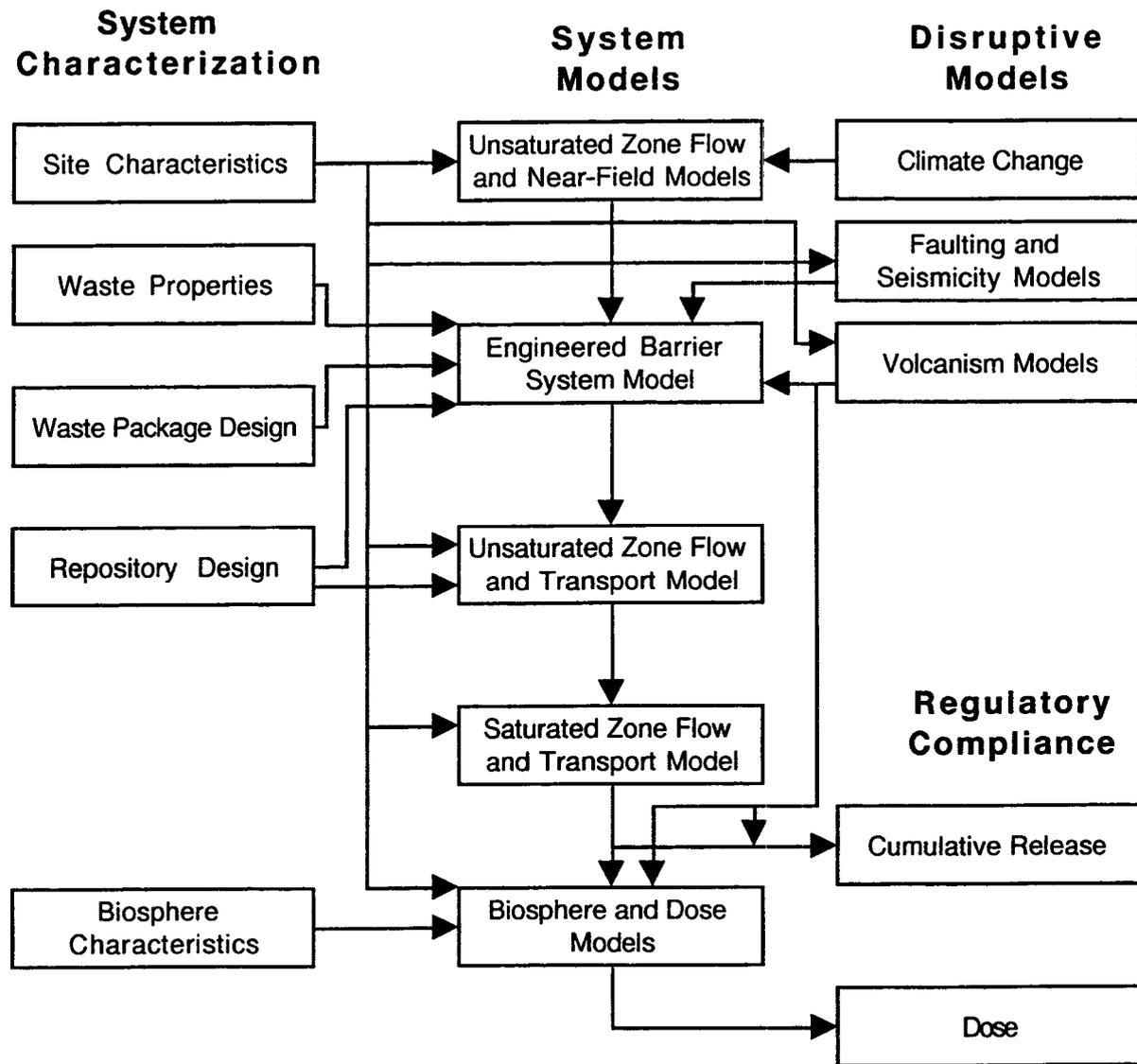


Figure 1. Overall TSPA flow diagram

- groundwater flow from the ground surface to the proposed repository,
- near-field THMC environment of the engineered barrier system (EBS),
- corrosion and other anticipated failure mechanisms of the EBS containment,
- release of radionuclides from the EBS into geologic setting,
- groundwater flow and radionuclide transport in the unsaturated zone below the proposed repository and into the saturated zone,
- groundwater flow and radionuclide transport in the saturated zone below the proposed repository to a compliance point (CP) or boundary, and
- transport of radionuclides in the biosphere through the groundwater pathway that leads to dose to humans.

The disruptive system has faulting, seismicity, volcanism and climate change that cause earlier failures of EBS containment. In the case of volcanism, radionuclides may be released directly into the accessible environment and at the CP through the ground surface pathway.

A number of utility modules will be used to provide general data and generic capabilities that more than one module may need. For example, the initial radionuclide inventory will be calculated in a module so this information can be provided to other modules.

The TPA code will control the spatial discretization of the proposed repository (i.e., number of SAs), the distance from the proposed repository to the CP (e.g., 5, 20, 25, or 30 km), the temporal discretization scheme (e.g., output every 200 yr), and the TPI (e.g., 100,000 yr).

Major analysis improvements in the TPA code include the ability to

- increase or decrease the TPI
- evaluate finer time discretizations using nonuniform time steps
- evaluate finer repository spatial discretizations
- evaluate different areal mass loading
- calculate time-dependent dose rate at a CP
- calculate peak dose rate at a CP in the TPI
- evaluate dilution in saturated zone
- evaluate in-drift emplacement design
- add or remove sampled parameters

The TPA code Version 3.0 will also include:

- updated consequence models
- improved parameter importance analysis capabilities
- streamlined scope for consequence modules
- streamlined methodology for data transfer between consequence modules
- more flexible design to accommodate changes in consequence modules

## 4 USER INTERFACE AND DATA FLOW

The TPA code will be executed in file batch mode using one main input file: "tpa.inp." The TPA code reads data from this one input file only. The TPA code writes output data into a set of files for plotting or importance analyses. The file interfaces are described here.

### 4.1 INPUT TO TOTAL PERFORMANCE ASSESSMENT CODE

All of the input data for the TPA code is contained in the "tpa.inp" file. No other files/input will have an effect on the TPA executive (EXEC) calculations. Auxiliary files for data may be needed for consequence modules, however, these files should be "static" and the "tpa.inp" file should be used for parameter descriptions that change from run to run. The "tpa.inp" file contains all parameters necessary to describe the scenario and the number of realizations requested. The "tpa.inp" file starts with two comment lines that the analysts should use to describe the type of run being performed. These lines will be read as Character\*80 and echoed at the top of all output files.

### 4.2 FLOW OF DATA BETWEEN MODULES

EXEC controls data flow by passing data in the subroutine call statement to each module. EXEC does not use common blocks or disk files for data transfer between itself and consequence modules. Within a consequence module, common blocks or files can be used. EXEC does not permit that data be passed directly between consequence modules. Each module is to be called only by EXEC and not by other modules. For efficiency and implementation purposes, the modules can consist of more than one subroutine, may call TPA code utility subroutines (e.g., INVENT), or may call stand alone programs (e.g., NEFTRAN). But modules are to pass information only to EXEC to control the simulation process.

The overall sequence of execution and flow of data is shown in figure 2. Here, EXEC starts the simulation by reading the "tpa.inp" file through the READER routine. The READER module need only be called once during a run. Having determined the parameters that describe the system, the EXEC continues by calling component specific consequence modules. Some modules will only be called once during one realization, while others will be called many times. Modules being called once include SAMPLER, SZFT, DCAGW and DCAGS. The modules UZFLOW, NFENV, EBSFAIL, EBSREL, and UZFT will be called once for each SA for each realization. If disruptive scenarios are being analyzed, the FAULTING, SEISMO, VOLCANO, ASHPLUME, and DCAGS modules will be called directly by EXEC once during a realization. These disruptive modules will be used to either cause earlier failure of the EBS or provide a more direct pathway for radionuclides into the biosphere (e.g., VOLCANO through ASHPLUME). If desired in the future, disruptive scenarios can be designed or modified to affect groundwater flow and transport of radionuclides.

The computational scheme for the TPA code is shown in figure 3. Each module is called by the EXEC module with some inputs and then returns some outputs to the EXEC through the call statement. Consequence modules do not call each other directly. There is a clear expectation of inputs and outputs between the EXEC and each module. In most cases, the output of one module is the input to the next module. The EXEC has two main loops for the number of realizations and the number of SAs.

The utility modules and consequence modules are discussed next.

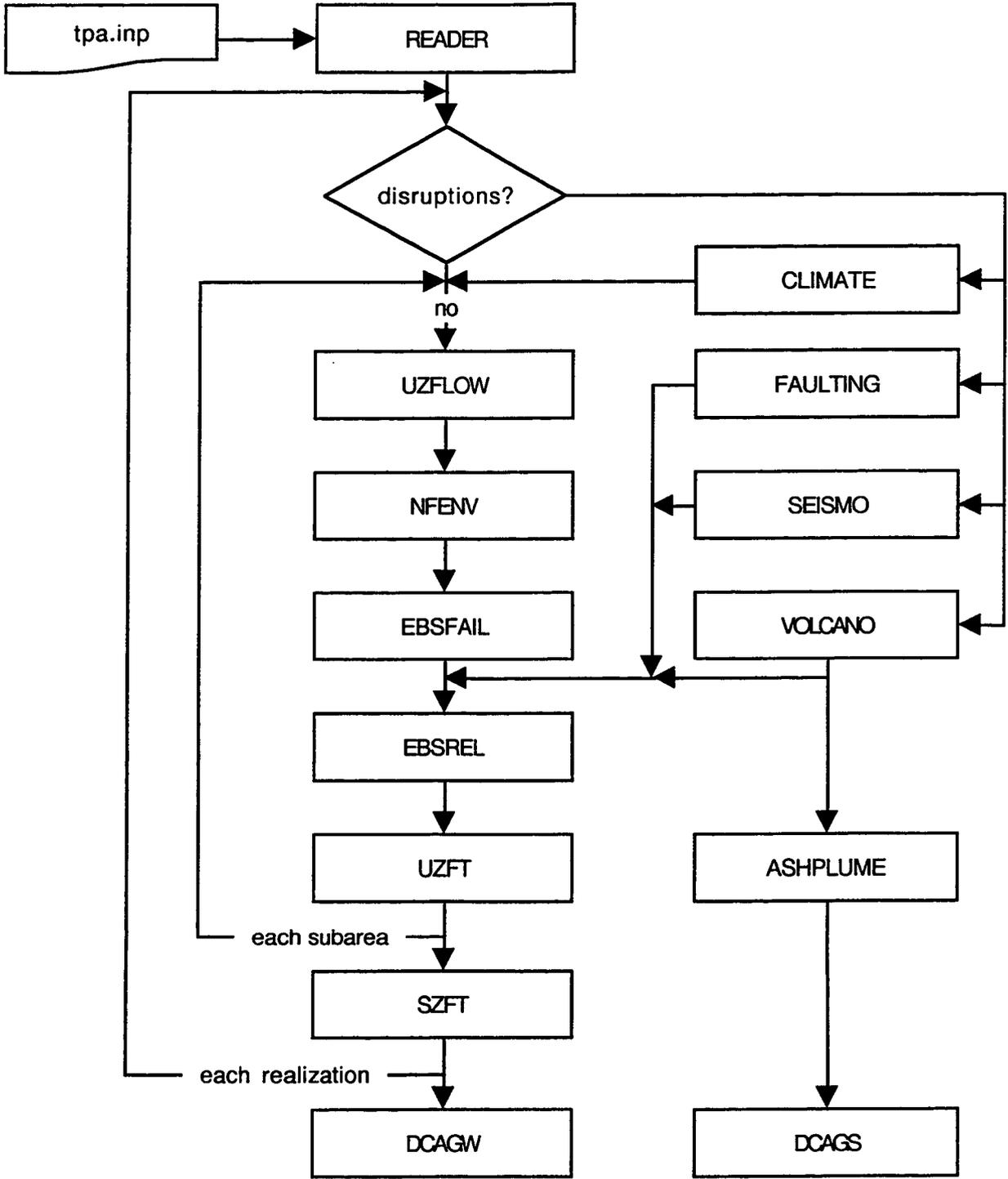


Figure 2. TPA flow diagram

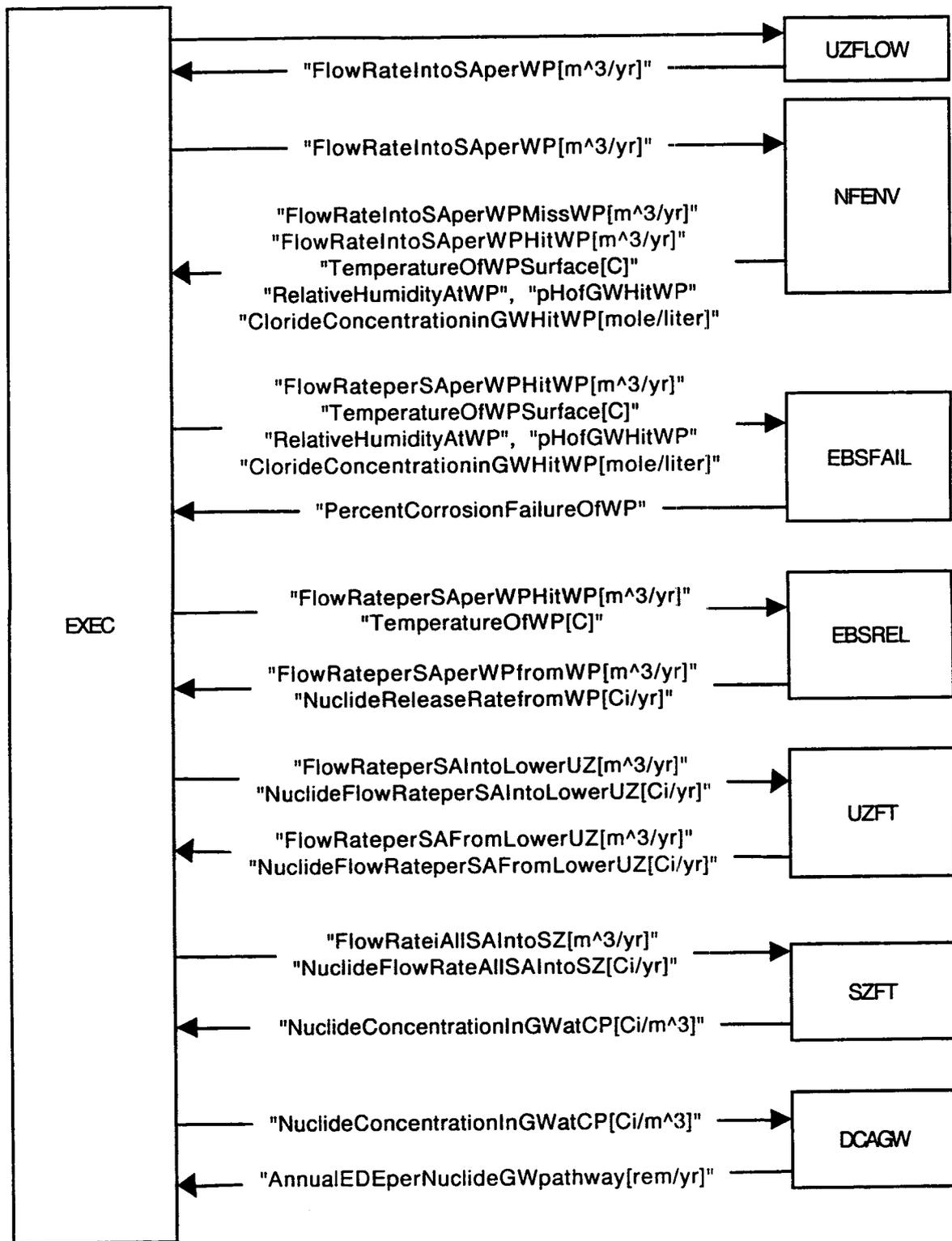


Figure 3. Main input/output associated with base-case flow and transport

## **4.3 UTILITY MODULES**

### **4.3.1 Reader**

READER is a utility module that preprocesses the data from the "tpa.inp" file. This is the only subroutine that reads the "tpa.inp" file. This module is similar to that already existing in the TPA code Version 2.0. The "tpa.inp" file will contain data specific for the TPA code execution as well as all probability distribution function (PDF) definitions for parameters that will be provided to the consequence modules.

### **4.3.2 Sampled Parameter**

This module dynamically stores and retrieves information associated with parameter probability density functions (PDFs). PDFs are read from the "tpa.inp" file during run time. The number of distributions includes: CONSTANT, UNIFORM, LOGUNIFORM, NORMAL, LOGNORMAL, BETA, LOGBETA, TRIANGULAR, LOGTRIANGULAR, and EXPONENTIAL. In addition joint PDFs relating two parameters will be supported using a correlation matrix approach. All PDFs will be sampled for each of the realizations required in the simulation.

### **4.3.3 Invent**

INVENT is a utility module that allows centralized computation and storage of radionuclide inventory data. This module is a set of subroutines based on the subroutines described in Lozano et al. (1994). The subroutines provide the inventory (in Ci/MTU) of 43 radionuclides for times to one million years.

### **4.3.4 Subarea**

SUBAREA is a utility module for the storage and retrieval of repository SA information. The database is created once in the READER module and then the information will be available to all other modules. The consequence modules can acquire information about the SA discretization, but not change the information.

## **4.4 CONSEQUENCE MODULES**

The main consequence modules in the TPA code are UZFLOW, NFENV, EBSFAIL, FAULTING, SEISMO, VOLCANO, EBSREL, UZFT, SZFT, ASHPLUME, DCAGW, and DCAGS. These modules will interface with EXEC using a subroutine call statement. The parameters and arrays being passed in the call statement are negotiated between the EXEC and each of the modules. Consistent with the software design principle of procedural abstraction, the EXEC does not need to know how the calculations are performed in the consequence modules. There are at least four ways of doing the calculations: abstraction of results (i.e., table look-up), abstraction of models, incorporation of the main calculational routines from an existing code, or spawning an independent process that executes a stand alone code. Previously, the EXEC explicitly called routines and spawned processes. In the new EXEC, main consequence modules are called directly and the I/O between EXEC and modules need not be changed in the future if the implementation in the module changes.

Because the TPA code simulates the time-dependent response of the proposed repository, it decides the overall time discretization to be used for all consequence modules. The time discretization is intended to synchronize input and output between modules and should not be confused with timesteps used in solving transient problems in various modules. The time discretization can and will often be nonuniform, especially for simulations out to hundreds of thousands of years. An example of a time discretization is {0, 10, 25, 50, 75, 100, 125, 150, 200, 250, 300, ..., 9,500, 10,000} yr. All of the time-dependent inputs and outputs for the consequence modules must be provided at these time steps.

The UZFLOW module will provide estimates of percolating flow rates into the near-field of the proposed repository. Separate flow rates will be estimated for each of the proposed repository SAs. For example, if six SAs are used for the proposed repository, then UZFLOW will be called six times to provide estimates of flow into the near-field. The flow rates are time-dependent and need to be predicted for the times provided by the TPA code. This module may account for long-term trends that affect percolation (e.g., climate changes) or short-term changes (e.g., abnormal wet period). Output data from the UZFLOW module will be input to the NFENV module.

The NFENV module will provide estimates of near-field conditions for each proposed repository SA. The NFENV module should account for the location of the SA (interior or edge regions). The output of the NFENV module is the near-field rock temperature, WP surface temperature, spent fuel temperature, relative humidity at the WP, flow rates into the EBS, and geochemical condition of groundwater flowing into the EBS. All of this output will be time-dependent. These data will be provided to EBSFAIL module.

The EBSFAIL module uses the output of the NFENV module to predict failure of the EBS to contain waste. Failure can be the result of corrosion, as well as other anticipated causes. Examples of anticipated causes of failure include initial defects, thermal-induced stresses in the WP, and anticipated seismic activity. A separate SEISMO module will also evaluate the consequences of seismic activity. At this time, EBSFAIL should account for the numerous small magnitude events while SEISMO should evaluate low-probability, high-consequence events. Possibly, EBSFAIL should evaluate activity with recurrence intervals of up to 500 – 1,000 yr, and SEISMO should evaluate stronger events that have longer recurrence intervals. EBSFAIL and SEISMO analysts need to negotiate this detail. Low-probability, high-consequence disruptive causes of failure are not considered in EBSFAIL. The primary output of EBSFAIL will be a time-dependent fraction of EBS failure to contain the waste. The fraction may start at a nonzero value due to initial defects and may not reach 100 percent within maximum simulation time.

The FAULTING, SEISMO, and VOLCANO modules each predict failure of the EBS due to disruptive processes and events or additional mechanical loads for the WP that accelerate failure of the EBS. These modules are called only for disruptive scenarios and not for the base-case scenario. Each module generates a time-dependent failure curve for WPs in each realization. These failures are combined by the TPA code with the EBSFAIL failures to have an overall percent failure.

The EBSREL module predicts the transient release rate [Ci/yr] of each radionuclide per WP in the SA. The radionuclides are released from the EBS and into the lower unsaturated zone region that extends from below the proposed repository to above the water table.

The UZFT module predicts release [Ci/yr] of each radionuclide from the unsaturated zone into the saturated zone. The module simulates gravity-driven percolating flow and radionuclide transport in the fractured, stratified hydrogeology. The module accounts for the retardation of nuclides. The sum of the releases from all SAs is then provided to the SZFT module.

The SZFT module predicts the transient groundwater source [Ci/m<sup>3</sup>] of each radionuclide at the CP which may be a well located 5, 20, 25 or 30 km away from the proposed repository. The module accounts for the retardation of nuclides, plume dilution, and dilution due to the pumping rate at the well.

The ASHPLUME module provides an extra pathway for radionuclides to be transported into the biosphere. This pathway is due to extrusive volcanic events that entrain waste in the magma and spread the waste in the volcanic ash plume. After VOLCANO is called, ASHPLUME will be called to evaluate this pathway for waste.

The DCAGW module simulates the biosphere and computes dose rates [rem/yr] from the groundwater pathway. DCAGW uses the output of SZFT. The DCAGW module is based on the GENII code which has been applied to YM biosphere conditions.

The DCAGS module simulates the biosphere and computes annual doses [rem/yr] from ground surface pathways. DCAGS uses output from ASHPLUME. The DCAGS module is based on the GENII code which has been applied to YM biosphere conditions.

#### **4.5 OUTPUT FROM TOTAL PERFORMANCE ASSESSMENT CODE**

The TPA code will generate results that can be used in importance analyses and in assessment of proposed repository compliance with either dose, risk, or release-based standards. The output files generated are listed here:

sp.dat = input values sampled for each R from PDFs described in "tpa.inp"

mv.dat = constant (e.g., not time- or nuclide-dependent) module variables for each module for each R/SA that will be used for importance analyses

uzflow.dat = output values for each R/SA from the UZFLOW module

nfenv.dat = output values for each R/SA from the NFENV module

ebsfail.dat = output values for each R/SA from the EBSFAIL module

ebsrel.dat = output values for each R/SA from the EBSREL module

uzft.dat = output values for each R/SA from the UZFT module

szft.dat = output values for each R from the SZFT module

dcagw.dat = output values for each R from the DCAGW module

dcags.dat = output values for each R from the DCAGS module

doseavg.dat = annual effective dose equivalent at CP

ccdf.dat = data used to generated complementary cumulative distribution function for Environmental Protection Agency normalized release to accessible environment located at 5 km from the proposed repository over a 10,000 yr time period

The first two lines of any output file will echo the first two lines of the "tpa.inp" file. The third line of any output file will provide the version number of the TPA code being used and the time and date of the run. Data will follow on subsequent lines depending on the specific file.

## 5 PROGRAMMING LANGUAGES

The TPA code is written in FORTRAN 77 as implemented in the SUN SPARCompiler, Version 2.0. The length of variable names will not be restricted to seven or less characters, but will be as long as needed to readily identify the variable. In addition, some compiler specific calls for date and time will be used. Although not recommended, modules can be written in languages other than FORTRAN 77 or can be standalone computer programs. In these cases, the responsible programmers must provide a FORTRAN 77 interface subprogram consistent with the TPA-module interface described in the previous section.

## 6 HARDWARE PLATFORMS

The TPA code will be developed for execution on SUN machines using the UNIX operating systems. The code will be designed such that it will also run on other operating systems to the extent practical, such as the CRAY computer.

## 7 GRAPHICS OUTPUT

No special graphics devices will be supported. Output will be in the form of ASCII files written in a format that can be read by spreadsheet programs, analysis software, and plotting packages.

## 8 PRE AND POST-PROCESSOR

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

## 9 REFERENCES

- Lozano, A.S., H. Karimi, J.P. Cornelius, R.D. Manetufel, and R.W. Janetzke. 1994. *INVENT: A module for Calculation of Radionuclide Inventories, Software description, and User Guide*, CNWRA 94-016. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.
- Sagar, B., and R.W. Janetzke. 1993. *Total-System Performance Assessment (TPA) Computer Code: Description of Executive Module, Version 2.0*. CNWRA 93-017. San Antonio, TX: Center For Nuclear Waste Regulatory Analyses.ames

## **APPENDIX B**

**EXAMPLE INPUT FILE: *tpa.inp***

## Example Input File: *tpa.inp*

```
**
title
  This is the test file for TPA 3.0beta version
  which is being supplied to NRC on February 26, 1997.
**
iflag
Volcanism disruptive scenario flag (yes=1, no=0)
0
**
iflag
Faulting disruptive scenario flag (yes=1, no=0)
0
**
iflag
Seismic disruptive scenario flag (yes=1, no=0)
0
**
** Number and Location Of SubAreas[m] Based On Fig3.4-1 in TSPA95
subarea
1
  ZONE T="ONE RECTANGULAR ZONE SUBAREA", F=POINT
    547400.0    4076200.0
    548600.0    4076200.0
    548600.0    4079040.0
    547400.0    4079040.0
    547400.0    4076200.0
**
**6
** ZONE T="Upper Block-NW",F=POINT
**   547615.8    4079673.8
**   548118.7    4079465.4
**   548033.3    4079140.2
**   547460.1    4079328.2
**   547615.8    4079673.8
** ZONE T="Upper Block-NE",F=POINT
**   548118.7    4079465.4
**   548621.6    4079257.1
**   548606.6    4078952.2
**   548033.3    4079140.2
**   548118.7    4079465.4
** ZONE T="Upper Block-W",F=POINT
**   547460.1    4079328.2
**   548033.3    4079140.2
**   547933.5    4077770.0
**   547323.9    4077962.2
**   547460.1    4079328.2
** ZONE T="Upper Block-E",F=POINT
**   548033.3    4079140.2
```

```

**      548606.6      4078952.2
**      548543.1      4077577.8
**      547933.5      4077770.0
**      548033.3      4079140.2
** ZONE T="Upper Block-SW",F=POINT
**      547323.9      4077962.2
**      547933.5      4077770.0
**      547997.4      4076342.2
**      547660.3      4076451.5
**      547323.9      4077962.2
** ZONE T="Upper Block-SE",F=POINT
**      547933.5      4077770.0
**      548543.1      4077577.8
**      548335.5      4076236.8
**      547997.4      4076342.2
**      547933.5      4077770.0
**1
** ZONE T="ONE RECTANGULAR ZONE SUBAREA", F=POINT
**      547400.0      4076200.0
**      548600.0      4076200.0
**      548600.0      4079040.0
**      547400.0      4079040.0
**      547400.0      4076200.0
**
**ZONE T="Upper Block-NW7",F=POINT
**      547742.9      4080134.0
**      548197.2      4079988.0
**      548118.7      4079465.4
**      547615.8      4079673.8
**      547742.9      4080134.0
** ZONE T="Upper Block-NE8",F=POINT
**      548197.2      4079988.0
**      548650.6      4079842.8
**      548621.6      4079257.1
**      548118.7      4079465.4
**      548197.2      4079988.0
**
** Number and Names of Nuclides to be Tracked for Aqueous Pathway
**aqueousnuclides
**5
** Am243
** Np237
** I129
** Tc99
** Cl36
**
**Number and Names of Nuclides to be Tracked for Aqueous Pathway
aqueousnuclides
21
Cm246
U238

```

Cm245  
Am241  
Np237  
Am243  
Pu239  
Pu240  
U234  
Th230  
Ra226  
Pb210  
Cs137  
Cs135  
I129  
Tc99  
Ni59  
C14  
Se79  
Nb94  
Cl36  
\*\*\*\*  
constant  
SeedForRandomNumber  
188910405.0  
\*\*  
iflag  
LatinHypercubeSampling (yes=1, no=0)  
0  
\*\*  
iconstant  
NumberOfRealizations  
10  
\*\*  
constant  
MaximumTime[yr]  
1e4  
\*\*  
iconstant  
NumberOfTimeSteps  
201  
\*\*  
constant  
RatioOfLastToFirstTimeStep  
100.  
\*\*  
constant  
ArealMassLoading[MTU/acre]  
83.0  
\*\* 25.0  
\*\* 87.2  
\*\*  
constant

WastePackagePayload[MTU]  
 8.8  
 \*\*  
 constant  
 AgeOfWaste[yr]  
 26.0  
 \*\*  
 constant  
 BurnupOfWaste[GWd/MTU]  
 38.5  
 \*\*  
 \*\* Parameters used in NFENV  
 \*\*  
 constant  
 ElevationOfRepositoryHorizon[m]  
 1072.0  
 \*\*  
 \*\* Gives Avg. Repository Depth of 328 [m] below ground surface  
 \*\*  
 constant  
 ElevationOfGroundSurface[m]  
 1400.0  
 \*\*  
 uniform  
 GroundwaterPercolationRate[mm/yr]  
 0.5, 2.0  
 \*\*  
 constant  
 DistanceToCriticalGroup[km][should be 5 or 30]  
 5.0  
 \*\*  
 uniform  
 WellPumpingRateAtCriticalGroup[gal/day]  
 1.0e3, 1.0e4  
 \*\*  
 \*\* Parameters used in FAULTING  
 \*\*  
 uniform  
 TimeOfNextFaultingEventInRegionOfInterest[yr]  
 0.0, 10000.0  
 \*\*  
 constant  
 ThresholdDisplacementforFaultDisruptionOfWP[m]  
 0.25  
 \*\*  
 uniform  
 XLocationOfFaultingEventInRegionOfInterest[m]  
 545000.0, 551000.0  
 \*\*  
 uniform  
 YLocationOfFaultingEventInRegionOfInterest[m]

4076000.0, 4082000.0  
 \*\*  
 \*\* 25% of time Fault has NW orientation  
 \*\* 75% of time Fault has NE orientation  
 uniform  
 RNtoDetermineFaultOrientation  
 0.0, 1.0  
 \*\*  
 \*\* for NW, between 25deg and 40deg 90% of time  
 \*\* for NE, between -5deg and 25deg 90% of time  
 \*\* +- 1.65 standard deviation  
 \*\*  
 normal  
 NWFaultStrikeOrientationMeasuredfromNorthClockwise[degrees]  
 -46.5, -18.5  
 \*\*  
 normal  
 NEFaultStrikeOrientationMeasuredfromNorthClockwise[degrees]  
 -18.1, 38.1  
 \*\*  
 uniform  
 NWFaultTraceLength[m]  
 2000.0, 10000.0  
 \*\*  
 uniform  
 NEFaultTraceLength[m]  
 3000.0, 12000.0  
 \*\*  
 logbeta  
 NWFaultZoneWidth[m]  
 1.5, 3.0, 0.5, 275.0  
 \*\*  
 logbeta  
 NEFaultZoneWidth[m]  
 1.5, 3.0, 0.5, 365.0  
 \*\*  
 iuniform  
 NumberOfFaultSlipSurfaces  
 1, 4  
 \*\*  
 uniform  
 NWAmountOfLargestCredibleDisplacement[m]  
 0.045, 0.250  
 \*\*  
 uniform  
 NEAmountOfLargestCredibleDisplacement[m]  
 0.060, 0.450  
 \*\*  
 uniform  
 NWCumulativeDisplacementRate[mm/yr]  
 0.0, 0.00001

```

**
uniform
NECumulativeDisplacementRate[mm/yr]
4.0E-8, 7.0E-6
**
finiteexponential
TimeOfNextVolcanicEventInRegionOfInterest[yr]
1.0e-8, 100.0, 10000.0
**
uniform
XLocationInRegionOfInterest[m]
544000.0, 550000.0
**
uniform
YLocationInRegionOfInterest[m]
4076000.0, 4082000.0
**
uniform
RNtoDetermineIfExtrusiveOrIntrusiveVolcanicEvent
0.0, 1.0
**
constant
FractionOfTimeVolcanicEventsIsExtrusive
0.999
**
uniform
AngleOfVolcanicDikeMeasuredFromNorthClockwise[degrees]
0.0, 15.0
**
uniform
LengthOfVolcanicDike[m]
1000.0, 4000.0
**
uniform
WidthOfVolcanicDike[m]
1.0, 10.0
**
uniform
DiameterOfVolcanicCone[m]
10.0, 50.0
**
uniform
AverageUndisturbedInfiltration[mm/yr]
0.5, 2.0
**
**uniform
**RandomNumberToSelect1of125GENIIRealizations
**1, 125
**
**
iconstant

```

nsetUsedToPickTempRHDataSet

1  
\*\* 1 Time Twp nv80nb Tw nv80nb RH nv80nb  
\*\* 2 Time Twp nv80yb Tw nv80yb RH nv80yb  
\*\* 3 Time Twp cv80nb Tw cv80nb RH cv80nb  
\*\* 4 Time Twp cv80yb Tw cv80yb RH cv80yb  
\*\* 5 Time Twp nv40yb Tw nv40yb RH nv40yb  
\*\* 6 Time Twp nv40nb Tw nv40nb RH nv40nb  
\*\* 7 Time Twp cv40nb Tw cv40nb RH cv40nb  
\*\* 8 Time Twp cv40yb Tw cv40yb RH cv40yb  
\*\* 9 Time Twp nv20nb Tw nv20nb RH nv20nb  
\*\* 10 Time Twp nv20yb Tw nv20yb RH nv20yb  
\*\* 11 Time Twp cv20nb Tw cv20nb RH cv20nb  
\*\* 12 Time Twp cv20yb Tw cv20yb RH cv20yb  
\*\*

constant  
WPLength[m]  
5.682  
\*\*

constant  
WPDiameter[m]  
1.802  
\*\*

constant  
EmplacementDriftDiameter[m]  
5.0  
\*\*

constant  
WPSpacingAlongEmplacementDrift[m]  
19.0  
\*\*

constant  
ThermalGridSize[m]  
90.0  
\*\*

constant  
AmbientRepositoryTemperature[C]  
20.0  
\*\*

constant  
AverageGroundSurfaceTemperature[C]  
13.0  
\*\*

constant  
AverageGeothermalGradient[C/km]  
20.0  
\*\*

constant  
MassDensityofYMRock[kg/m^3]  
2580.0  
\*\*

constant  
SpecificHeatofYMRock[J/(kg-K)]  
840.0  
\*\*  
uniform  
ThermalConductivityofYMRock[W/(m-K)]  
1.8, 2.2  
\*\*  
constant  
EmissivityOfDriftWall[-]  
0.8  
\*\*  
constant  
EmissivityOfWastePackage[-]  
0.7  
\*\*  
constant  
ThermalConductivityOfFloor[W/(m-C)]  
0.6  
\*\*  
constant  
ThermalConductivityOfStagnantAir[W/(m-C)]  
0.030  
\*\*  
\*\* Use keff = 30 \* kair, from Manteufel (1996)  
constant  
EffectiveThermalConductivityOfUnbackfilledDrift[W/(m-C)]  
0.900  
\*\*  
constant  
TimeOfBackfillEmplaced[yr]  
100.0  
\*\*  
constant  
EffectiveThermalConductivityOfBackfill[W/(m-C)]  
0.60  
\*\*  
constant  
ThermalConductivityOfInnerStainlessSteelWall[W/m-C]  
15.0  
\*\*  
constant  
ThermalConductivityOfOuterCarbonSteelWall[W/m-C]  
50.0  
\*\*  
constant  
EffectiveThermalConductivityOfBasket&SFinWP[W/(m-C)]  
1.0  
\*\*  
constant  
InnerWPThickness[m]

0.02  
\*\*  
constant  
OuterWPThickness[m]  
0.1  
\*\*  
uniform  
CriticalRelativeHumidity  
0.60, 0.70  
\*\*  
constant  
ReferencepH  
9.0  
\*\*  
constant  
BoilingPointofWater[C]  
97.0  
\*\*  
constant  
ThicknessOfWaterFilm[m]  
0.002  
\*\*  
constant  
OuterWPBetaKineticsParameterforOxygen  
0.75  
\*\*  
constant  
OuterWPBetaKineticsParameterforWater  
0.50  
\*\*  
constant  
InnerWPBetaKineticsParameterforOxygen  
0.75  
\*\*  
constant  
InnerWPBetaKineticsParameterforWater  
0.50  
\*\*  
constant  
OuterWPRateConstantforOxygenReduction[coulomb m/mole/yr]  
3.8e12  
\*\*  
constant  
OuterWPRateConstantforWaterReduction[coulomb m/m^2/yr]  
1.6e-1  
\*\*  
constant  
InnerWPRateConstantforOxygenReduction[coulomb m/mole/yr]  
3.0e10  
\*\*  
constant

InnerWPRateConstantforWaterReduction[coulomb m/m<sup>2</sup>/yr]  
3.2  
\*\*  
constant  
OuterWPActivationEnergyforOxygenReduction[J/mole]  
37300.0  
\*\*  
constant  
OuterWPActivationEnergyforWaterReduction[J/mole]  
25000.0  
\*\*  
constant  
InnerWPActivationEnergyforOxygenReduction[J/mole]  
40000.0  
\*\*  
constant  
InnerWPActivationEnergyforWaterReduction[J/mole]  
25000.0  
\*\*  
constant  
PorosityOfScaleonWP  
1.0  
\*\*  
constant  
TortuosityforLiquidDiffusionofO2throughBoilerScale  
1.0  
\*\*  
constant  
FractionalCouplingStrength  
0.0  
\*\*  
constant  
MetalGrainRadius[micrometer]  
5.0  
\*\*  
constant  
GrainBoundaryThickness[micrometer]  
0.0007  
\*\*  
constant  
AboveBoilingChlorideConcentration[mole/liter]  
0.3  
\*\*  
constant  
BelowBoilingChlorideConcentration[mole/liter]  
0.003  
\*\*  
constant  
FractionOfFlowHittingWP  
0.05  
\*\*

\*\* Solubility PDFs recommended by  
\*\* R. Pabalan & D. Turner in 10/18/96 memo  
\*\*

uniform  
SolubilityAm[mole/liter]  
1.0e-10, 1.0e-6  
\*\*

triangular  
SolubilityNp[mole/liter]  
5.0e-6, 1.4e-4, 1.0e-2  
\*\*

constant  
Solubility\_I[mole/liter]  
1.0  
\*\*

constant  
SolubilityTc[mole/liter]  
1.0  
\*\*

constant  
SolubilityCl[mole/liter]  
1.0  
\*\*

constant  
Solubility\_C[mole/liter]  
1.0  
\*\*

triangular  
Solubility\_U[mole/liter]  
1.0e-8 3.2e-5 1.0e-2  
\*\*

uniform  
SolubilityCm[mole/liter]  
1.0e-10 1.0e-6  
\*\*

uniform  
SolubilityPu[mole/liter]  
1.0e-8 1.0e-6  
\*\*

loguniform  
SolubilityTh[mole/liter]  
1.0e-9 1.0e-3  
\*\*

triangular  
SolubilityRa[mole/liter]  
1.0e-9 1.0e-7 1.0e-5  
\*\*

triangular  
SolubilityPb[mole/liter]  
1.0e-8 3.1e-7 1.0e-5  
\*\*

```

constant
SolubilityCs[mole/liter]
1.0
**

triangular
SolubilityNi[mole/liter]
1.0e-6 1.9e-3 1.0e-1
**

constant
SolubilitySe[mole/liter]
1.0
**

loguniform
SolubilityNb[mole/liter]
1.0e-9 1.0e-7
**

**constant
**MaximumNuclideReleaseRatefromWP[1/yr]
**1.0e-5
**

hazardcurve
SeismicHazardCurveforSEISMO
3
0.10 100.0
0.30 1000.0
0.60 10000.0
** 6 // Number of Seismic bins
** 0.25 1000.0 // Min Peak Acceleration for Bin, Return Period [yr]
** 0.4 2500.0 // Min Peak Acceleration for Bin, Return Period [yr]
** 0.5 4000.0 // Min Peak Acceleration for Bin, Return Period [yr]
** 0.7 16000.0 // Min Peak Acceleration for Bin, Return Period [yr]
** 0.9 30000.0 // Min Peak Acceleration for Bin, Return Period [yr]
** 1.0 50000.0 // Min Peak Acceleration for Bin, Return Period [yr]
**

iuniform
RealizationforSEISMO
1, 4
**

constant
DefectiveFractionOfWP
0.00
**

constant
FunnelFactor
7.0
**

constant
FlowMultiplicationFactor
1.0
**
** See WastePackagePayload[MTU]

```

```

** constant
** MassOfSpentFuelinWP[kg]
** 2800.0 <<<--- why not 8800 kg ???
**
iconstant
IndexForIdentifyingLeachingRateModel
2
**
constant
DissolvedOxygenOverPressure[atm]
0.2
**
constant
CarbonateConcentration[moles/liter]
0.002
**
** Parameters for ASHPLUME
**
constant
DensityOfAirAtSTP[g/cm3]
0.00129
**
constant
ViscosityOfAirAtSTP[g/cm-s]
0.00018
**
constant
ConstantRelatingFallTimeToEddyDiffusivity[cm2/s5/2]
400.0
**
constant
MaximumParticleDiameterForParticleTransport[cm]
10.d0
**
**logtriangular
**ParticleSizeinAshPlume[mm]
**0.5, 55.0, 125.5
constant
MinimumFuelParticulateSize[cm]
0.01
**
constant
ModeFuelParticulateSize[cm]
0.1
**
constant
MaximumFuelParticulateSize[cm]
1.0
**
constant
MinimumAshDensityForVariationWithSize[g/cm3]

```

```
0.8
**
constant
MaximumAshDensityForVariationWithSize[g/cm3]
2.5
**
constant
MinimumAshLogdiameterForDensityVariation
-2.0
**
constant
MaximumAshLogdiameterForDensityVariation
-1.0
**
constant
ParticleShapeParameter
0.5
**
constant
IncorporationRatio
0.3
**
constant
WindDirection[degrees]
-90.
**
constant
WindSpeed[cm/s]
1000.0
**
constant
VolcanicEventDuration[s]
331785.0
**
constant
VolcanicEventPower[W]
2.2118d12
**
constant
VolcanicColumnConstantBeta
0.02
**
constant
AshMeanParticleLogDiameter[d in cm]
0.1
**
constant
AshParticleSizeDistributionStandardDeviation
0.4
**
** Parameters for ASHREMOVE
```

```

**
constant
RelativeRateOfBlanketRemoval[1/yr]
0.001
**
constant
FractionOfPrecipitationLostToEvapotranspiration
0.5
**
constant
FractionOfIrrigationLostToEvapotranspiration
0.5
**
constant
AnnualPrecipitation[m/yr]
0.15
**
constant
AnnualIrrigation[m/yr]
1.52
**
constant
FractionOfYearSoilsSaturatedDueToPrecipitation
0.054
**
constant
FractionOfYearSoilsSaturatedDueToIrrigation
0.2
**
constant
AshBulkDensity[g/cm3]
2.0
**
constant
AshVolumetricMoistureFractionAtSaturation
0.4
**
constant
TimeOfExtrusiveVolcanicEvent[yr]
3565.0
**
constant
DepthOfTheRootingZone[m]
0.15
**
constant
KdOfUraniumInVolcanicAsh[m3/kg]
35.0
**
constant
KdOfCuriumInVolcanicAsh[m3/kg]

```

4000.0  
\*\*  
constant  
KdOfPlutoniumInVolcanicAsh[m3/kg]  
550.0  
\*\*  
constant  
KdOfAmericiumInVolcanicAsh[m3/kg]  
1900.0  
\*\*  
constant  
KdOfThoriumInVolcanicAsh[m3/kg]  
3200.0  
\*\*  
constant  
KdOfRadiumInVolcanicAsh[m3/kg]  
500.0  
\*\*  
constant  
KdOfLeadInVolcanicAsh[m3/kg]  
270.0  
\*\*  
constant  
KdOfProtactiniumInVolcanicAsh[m3/kg]  
550.0  
\*\*  
constant  
KdOfActiniumInVolcanicAsh[m3/kg]  
450.0  
\*\*  
constant  
KdOfNeptuniumInVolcanicAsh[m3/kg]  
5.0  
\*\*  
constant  
KdOfSamariumInVolcanicAsh[m3/kg]  
245.0  
\*\*  
constant  
KdOfCesiumInVolcanicAsh[m3/kg]  
280.0  
\*\*  
constant  
KdOfIodineInVolcanicAsh[m3/kg]  
1.0  
\*\*  
constant  
KdOfTinInVolcanicAsh[m3/kg]  
130.0  
\*\*  
constant

KdOfSilverInVolcanicAsh[m3/kg]  
55.0  
\*\*  
constant  
KdOfPaladiumInVolcanicAsh[m3/kg]  
55.0  
\*\*  
constant  
KdOfTechnetiumInVolcanicAsh[m3/kg]  
0.1  
\*\*  
constant  
KdOfMolybdenumInVolcanicAsh[m3/kg]  
10.0  
\*\*  
constant  
KdOfNiobiumInVolcanicAsh[m3/kg]  
160.0  
\*\*  
constant  
KdOfZirconiumInVolcanicAsh[m3/kg]  
600.0  
\*\*  
constant  
KdOfStrontiumInVolcanicAsh[m3/kg]  
15.0  
\*\*  
constant  
KdOfSeleniumInVolcanicAsh[m3/kg]  
150.0  
\*\*  
constant  
KdOfNickelInVolcanicAsh[m3/kg]  
400.0  
\*\*  
constant  
KdOfChlorineInVolcanicAsh[m3/kg]  
0.0  
\*\*  
constant  
KdOfCarbonInVolcanicAsh[m3/kg]  
5.0  
\*\*  
constant  
SolubilityOfUraniumInVolcanicAsh[moles/liter]  
1.0  
\*\*  
constant  
SolubilityOfCuriumInVolcanicAsh[moles/liter]  
1.0  
\*\*

constant  
SolubilityOfPlutoniumInVolcanicAsh[moles/liter]  
1.0  
\*\*

constant  
SolubilityOfAmericiumInVolcanicAsh[moles/liter]  
1.0  
\*\*

constant  
SolubilityOfThoriumInVolcanicAsh[moles/liter]  
1.0  
\*\*

constant  
SolubilityOfRadiumInVolcanicAsh[moles/liter]  
1.0  
\*\*

constant  
SolubilityOfLeadInVolcanicAsh[moles/liter]  
1.0  
\*\*

constant  
SolubilityOfProtactiniumInVolcanicAsh[moles/liter]  
1.0  
\*\*

constant  
SolubilityOfActiniumInVolcanicAsh[moles/liter]  
1.0  
\*\*

constant  
SolubilityOfNeptuniumInVolcanicAsh[moles/liter]  
1.0  
\*\*

constant  
SolubilityOfSamariumInVolcanicAsh[moles/liter]  
1.0  
\*\*

constant  
SolubilityOfCesiumInVolcanicAsh[moles/liter]  
1.0  
\*\*

constant  
SolubilityOfIodineInVolcanicAsh[moles/liter]  
1.0  
\*\*

constant  
SolubilityOfTinInVolcanicAsh[moles/liter]  
1.0  
\*\*

constant  
SolubilityOfSilverInVolcanicAsh[moles/liter]  
1.0

```

**
constant
SolubilityOfPaladiumInVolcanicAsh[moles/liter]
1.0
**
constant
SolubilityOfTechnetiumInVolcanicAsh[moles/liter]
1.0
**
constant
SolubilityOfMolybdenumInVolcanicAsh[moles/liter]
1.0
**
constant
SolubilityOfNiobiumInVolcanicAsh[moles/liter]
1.0
**
constant
SolubilityOfZirconiumInVolcanicAsh[moles/liter]
1.0
**
constant
SolubilityOfStrontiumInVolcanicAsh[moles/liter]
1.0
**
constant
SolubilityOfSeleniumInVolcanicAsh[moles/liter]
1.0
**
constant
SolubilityOfNickelInVolcanicAsh[moles/liter]
1.0
**
constant
SolubilityOfChlorineInVolcanicAsh[moles/liter]
1.0
**
constant
SolubilityOfCarbonInVolcanicAsh[moles/liter]
1.0
**
** Parameters for SZFT & UZFT
**
**
constant
WaterTableElevation[m]
880.0
**
constant
SaturatedZonePressureGradient_LAF_

```

3.0E-4  
\*\*  
constant  
SaturatedZonePressureGradient\_UAF\_  
3.0E-4  
\*\*  
constant  
SaturatedZonePressureGradient\_AV\_\_  
3.0E-4  
\*\*  
constant  
SaturatedZonePressureGradient\_tac\_  
3.0E-4  
\*\*  
constant  
SaturatedZonePressureGradient\_TCw\_  
3.0E-4  
\*\*  
constant  
SaturatedZonePressureGradient\_PTn\_  
3.0E-4  
\*\*  
constant  
SaturatedZonePressureGradient\_TSw\_  
3.0E-4  
\*\*  
constant  
SaturatedZonePressureGradient\_TSv\_  
3.0E-4  
\*\*  
constant  
SaturatedZonePressureGradient\_CHnv  
3.0E-4  
\*\*  
constant  
SaturatedZonePressureGradient\_CHnz  
3.0E-4  
\*\*  
constant  
SaturatedZonePressureGradient\_PPw\_  
3.0E-4  
\*\*  
constant  
SaturatedZonePressureGradient\_UCF\_  
3.0E-4  
\*\*  
constant  
SaturatedZonePressureGradient\_BFw\_  
3.0E-4  
\*\*  
constant

```

SaturatedZonePressureGradient_TR__
3.0E-4
**
constant
SaturatedZonePressureGradient_MCF_
3.0E-4
**
constant
MatrixLongitudinalDispersivity[FractionOfLayer]
0.1
**
constant
FractureLongitudinalDispersivity[FractionOfLayer]
0.01
**
constant
MatrixKD_LAF_Am
1.0
**
constant
MatrixKD_UAF_Am
1.0
**
constant
MatrixKD_AV__Am
1.0
**
constant
MatrixKD_tac_Am
1.0
**
constant
MatrixKD_TCw_Am
1.0
**
constant
MatrixKD_PTn_Am
1.0
**
loguniform
MatrixKD_TSw_Am
0.8 8.0
**
loguniform
MatrixKD_TSv_Am
0.8 8.0
**
loguniform
MatrixKD_CHnvAm
0.8 8.0
**

```

loguniform  
MatrixKD\_CHnzAm  
0.17 17.  
\*\*  
loguniform  
MatrixKD\_PPw\_Am  
0.45 45.0  
\*\*  
loguniform  
MatrixKD\_UCF\_Am  
0.136 13.6  
\*\*  
loguniform  
MatrixKD\_BFw\_Am  
0.014 1.4  
\*\*  
loguniform  
MatrixKD\_TR\_\_Am  
0.014 1.4  
\*\*  
loguniform  
MatrixKD\_MCF\_Am  
0.136 13.6  
\*\*  
constant  
MatrixKD\_LAF\_Np  
1.0  
\*\*  
constant  
MatrixKD\_UAF\_Np  
1.0  
\*\*  
constant  
MatrixKD\_AV\_\_Np  
1.0  
\*\*  
constant  
MatrixKD\_tac\_Np  
1.0  
\*\*  
constant  
MatrixKD\_TCw\_Np  
1.0  
\*\*  
constant  
MatrixKD\_PTn\_Np  
1.0  
\*\*  
loguniform  
MatrixKD\_TSw\_Np  
4.e-4 4.e-2

```
**
loguniform
MatrixKD_TSv_Np
4.e-4  4.e-2
**
loguniform
MatrixKD_CHnvNp
4.e-4  4.e-2
**
loguniform
MatrixKD_CHnzNp
2.7e-4  2.7e-2
**
loguniform
MatrixKD_PPw_Np
0.00051  0.051
**
loguniform
MatrixKD_UCF_Np
0.00022  0.022
**
loguniform
MatrixKD_BFw_Np
0.00051  0.051
**
loguniform
MatrixKD_TR__Np
0.00051  0.051
**
loguniform
MatrixKD_MCF_Np
0.00022  0.022
**
constant
MatrixKD_LAF_I
0.0
**
constant
MatrixKD_UAF_I
0.0
**
constant
MatrixKD_AV__I
0.0
**
constant
MatrixKD_tac_I
0.0
**
constant
MatrixKD_TCw_I
```

0.0  
\*\*  
constant  
MatrixKD\_PFn\_I  
0.0  
\*\*  
constant  
MatrixKD\_TSw\_I  
0.0  
\*\*  
constant  
MatrixKD\_TSv\_I  
0.0  
\*\*  
constant  
MatrixKD\_CHvI  
0.0  
\*\*  
constant  
MatrixKD\_CHzI  
0.0  
\*\*  
constant  
MatrixKD\_PPw\_I  
0.0  
\*\*  
constant  
MatrixKD\_UCF\_I  
0.0  
\*\*  
constant  
MatrixKD\_BFw\_I  
0.0  
\*\*  
constant  
MatrixKD\_TR\_I  
0.0  
\*\*  
constant  
MatrixKD\_MCF\_I  
0.0  
\*\*  
constant  
MatrixKD\_LAF\_Tc  
0.0  
\*\*  
constant  
MatrixKD\_UAF\_Tc  
0.0  
\*\*  
constant

MatrixKD\_AV\_Tc  
0.0  
\*\*  
constant  
MatrixKD\_tac\_Tc  
0.0  
\*\*  
constant  
MatrixKD\_TCw\_Tc  
0.0  
\*\*  
constant  
MatrixKD\_PTn\_Tc  
0.0  
\*\*  
constant  
MatrixKD\_TSw\_Tc  
0.0  
\*\*  
constant  
MatrixKD\_Tsv\_Tc  
0.0  
\*\*  
constant  
MatrixKD\_CHnvTc  
0.0  
\*\*  
constant  
MatrixKD\_CHnzTc  
0.0  
\*\*  
constant  
MatrixKD\_PPw\_Tc  
0.0  
\*\*  
constant  
MatrixKD\_UCF\_Tc  
0.0  
\*\*  
constant  
MatrixKD\_BFw\_Tc  
0.0  
\*\*  
constant  
MatrixKD\_TR\_Tc  
0.0  
\*\*  
constant  
MatrixKD\_MCF\_Tc  
0.0  
\*\*

```
constant
MatrixKD_LAF_CI
0.0
**
constant
MatrixKD_UAF_CI
0.0
**
constant
MatrixKD_AV_CI
0.0
**
constant
MatrixKD_tac_CI
0.0
**
constant
MatrixKD_TCw_CI
0.0
**
constant
MatrixKD_PTn_CI
0.0
**
constant
MatrixKD_TSw_CI
0.0
**
constant
MatrixKD_Tsv_CI
0.0
**
constant
MatrixKD_CHnvCI
0.0
**
constant
MatrixKD_CHnzCI
0.0
**
constant
MatrixKD_PPw_CI
0.0
**
constant
MatrixKD_UCF_CI
0.0
**
constant
MatrixKD_BFw_CI
0.0
```

```
**
constant
MatrixKD_TR_Cl
0.0
**
constant
MatrixKD_MCF_Cl
0.0
**
constant
MatrixKD_LAF_Cm
1.0
**
constant
MatrixKD_UAF_Cm
1.0
**
constant
MatrixKD_AV_Cm
1.0
**
constant
MatrixKD_tac_Cm
1.0
**
constant
MatrixKD_TCw_Cm
1.0
**
constant
MatrixKD_PTn_Cm
1.0
**
loguniform
MatrixKD_TSw_Cm
4.e-2 4.5
**
loguniform
MatrixKD_Tsv_Cm
4.e-2 4.5
**
loguniform
MatrixKD_CHnvCm
0.328 32.0
**
loguniform
MatrixKD_CHnzCm
0.16 16.0
**
loguniform
MatrixKD_PPw_Cm
```

0.116 11.6  
\*\*  
loguniform  
MatrixKD\_UCF\_Cm  
0.132 13.2  
\*\*  
loguniform  
MatrixKD\_BFw\_Cm  
0.12 12.0  
\*\*  
loguniform  
MatrixKD\_TR\_Cm  
0.12 12.0  
\*\*  
loguniform  
MatrixKD\_MCF\_Cm  
0.132 13.2  
\*\*  
constant  
MatrixKD\_LAF\_U  
1.0  
\*\*  
constant  
MatrixKD\_UAF\_U  
1.0  
\*\*  
constant  
MatrixKD\_AV\_\_U  
1.0  
\*\*  
constant  
MatrixKD\_tac\_U  
1.0  
\*\*  
constant  
MatrixKD\_TCw\_U  
1.0  
\*\*  
constant  
MatrixKD\_PTn\_U  
1.0  
\*\*  
loguniform  
MatrixKD\_TSw\_U  
2.e-5 2.e-3  
\*\*  
loguniform  
MatrixKD\_TSv\_U  
2.e-5 2.e-3  
\*\*  
loguniform

MatrixKD\_CHnvU  
2.e-3 0.2  
\*\*  
loguniform  
MatrixKD\_CHnzU  
1.e-4 1.e-2  
\*\*  
constant  
MatrixKD\_PPw\_U  
0.0  
\*\*  
loguniform  
MatrixKD\_UCF\_U  
8.0e-5 8.0e-3  
\*\*  
loguniform  
MatrixKD\_BFw\_U  
0.0002 0.02  
\*\*  
loguniform  
MatrixKD\_TR\_\_U  
0.0002 0.02  
\*\*  
loguniform  
MatrixKD\_MCF\_U  
8.0e-5 8.0e-3  
\*\*  
constant  
MatrixKD\_LAF\_Pu  
1.0  
\*\*  
constant  
MatrixKD\_UAF\_Pu  
1.0  
\*\*  
constant  
MatrixKD\_AV\_\_Pu  
1.0  
\*\*  
constant  
MatrixKD\_tac\_Pu  
1.0  
\*\*  
constant  
MatrixKD\_TCw\_Pu  
1.0  
\*\*  
constant  
MatrixKD\_PTn\_Pu  
1.0  
\*\*

loguniform  
MatrixKD\_TSw\_Pu  
.017 1.7  
\*\*

loguniform  
MatrixKD\_TV\_Pu  
.017 1.7  
\*\*

loguniform  
MatrixKD\_CHnvPu  
.017 1.7  
\*\*

loguniform  
MatrixKD\_CHnzPu  
0.0066 .66  
\*\*

loguniform  
MatrixKD\_PPw\_Pu  
6.6e-3 .66  
\*\*

loguniform  
MatrixKD\_UCF\_Pu  
0.0053 0.53  
\*\*

loguniform  
MatrixKD\_BFw\_Pu  
0.0094 0.94  
\*\*

loguniform  
MatrixKD\_TR\_Pu  
0.0094 0.94  
\*\*

loguniform  
MatrixKD\_MCF\_Pu  
0.0053 0.53  
\*\*

constant  
MatrixKD\_LAF\_Th  
1.0  
\*\*

constant  
MatrixKD\_UAF\_Th  
1.0  
\*\*

constant  
MatrixKD\_AV\_Th  
1.0  
\*\*

constant  
MatrixKD\_tac\_Th  
1.0

```
**
constant
MatrixKD_TCw_Th
1.0
**
constant
MatrixKD_PTn_Th
1.0
**
loguniform
MatrixKD_TSw_Th
0.004 0.4
**
loguniform
MatrixKD_Tsv_Th
0.004 0.4
**
loguniform
MatrixKD_CHnvTh
0.03 3.40
**
loguniform
MatrixKD_CHnzTh
0.017 1.7
**
loguniform
MatrixKD_PPw_Th
0.012 1.2
**
loguniform
MatrixKD_UCF_Th
0.014 1.4
**
loguniform
MatrixKD_BFw_Th
0.013 1.3
**
loguniform
MatrixKD_TR__Th
0.013 1.3
**
loguniform
MatrixKD_MCF_Th
0.014 1.4
**
constant
MatrixKD_LAF_Ra
1.0
**
constant
MatrixKD_UAF_Ra
```

```
1.0
**
constant
MatrixKD_AV__Ra
1.0
**
constant
MatrixKD_tac_Ra
1.0
**
constant
MatrixKD_TCw_Ra
1.0
**
constant
MatrixKD_PTn_Ra
1.0
**
loguniform
MatrixKD_TSw_Ra
0.15 15.
**
loguniform
MatrixKD_TSv_Ra
0.15 15.
**
loguniform
MatrixKD_CHnvRa
0.15 15.
**
loguniform
MatrixKD_CHnzRa
0.15 15.
**
loguniform
MatrixKD_PPw_Ra
0.15 15.0
**
loguniform
MatrixKD_UCF_Ra
0.12 12.0
**
loguniform
MatrixKD_BFw_Ra
0.5 50.0
**
loguniform
MatrixKD_TR__Ra
0.5 50.0
**
loguniform
```

MatrixKD\_MCF\_Ra  
0.12 12.0  
\*\*  
constant  
MatrixKD\_LAF\_Pb  
1.0  
\*\*  
constant  
MatrixKD\_UAF\_Pb  
1.0  
\*\*  
constant  
MatrixKD\_AV\_\_Pb  
1.0  
\*\*  
constant  
MatrixKD\_tac\_Pb  
1.0  
\*\*  
constant  
MatrixKD\_TCw\_Pb  
1.0  
\*\*  
constant  
MatrixKD\_PTn\_Pb  
1.0  
\*\*  
loguniform  
MatrixKD\_TSw\_Pb  
6.8e-4 .068  
\*\*  
loguniform  
MatrixKD\_Tsv\_Pb  
6.8e-4 .068  
\*\*  
loguniform  
MatrixKD\_CHnvPb  
0.0049 0.49  
\*\*  
loguniform  
MatrixKD\_CHnzPb  
0.0025 0.25  
\*\*  
loguniform  
MatrixKD\_PPw\_Pb  
0.0017 0.17  
\*\*  
loguniform  
MatrixKD\_UCF\_Pb  
0.0020 0.20  
\*\*

loguniform  
MatrixKD\_BFw\_Pb  
0.0018 0.18  
\*\*

loguniform  
MatrixKD\_TR\_\_Pb  
0.0018 0.18  
\*\*

loguniform  
MatrixKD\_MCF\_Pb  
0.002 0.20  
\*\*

constant  
MatrixKD\_LAF\_Cs  
1.0  
\*\*

constant  
MatrixKD\_UAF\_Cs  
1.0  
\*\*

constant  
MatrixKD\_AV\_\_Cs  
1.0  
\*\*

constant  
MatrixKD\_tac\_Cs  
1.0  
\*\*

constant  
MatrixKD\_TCw\_Cs  
1.0  
\*\*

constant  
MatrixKD\_PTn\_Cs  
1.0  
\*\*

loguniform  
MatrixKD\_TSw\_Cs  
0.036 3.6  
\*\*

loguniform  
MatrixKD\_TSv\_Cs  
0.036 3.6  
\*\*

loguniform  
MatrixKD\_CHnvCs  
0.024 2.4  
\*\*

loguniform  
MatrixKD\_CHnzCs  
2.2 220.

\*\*  
loguniform  
MatrixKD\_PPw\_Cs  
0.22 22.0  
\*\*  
loguniform  
MatrixKD\_UCF\_Cs  
1.76 176.0  
\*\*  
loguniform  
MatrixKD\_BFw\_Cs  
0.32 32.0  
\*\*  
loguniform  
MatrixKD\_TR\_Cs  
0.32 32.0  
\*\*  
loguniform  
MatrixKD\_MCF\_Cs  
1.76 176.0  
\*\*  
constant  
MatrixKD\_LAF\_Ni  
1.0  
\*\*  
constant  
MatrixKD\_UAF\_Ni  
1.0  
\*\*  
constant  
MatrixKD\_AV\_\_Ni  
1.0  
\*\*  
constant  
MatrixKD\_tac\_Ni  
1.0  
\*\*  
constant  
MatrixKD\_TCw\_Ni  
1.0  
\*\*  
constant  
MatrixKD\_PTn\_Ni  
1.0  
\*\*  
loguniform  
MatrixKD\_TSw\_Ni  
3.7e-4 0.037  
\*\*  
loguniform  
MatrixKD\_Tsv\_Ni

3.7e-4 0.037  
\*\*  
loguniform  
MatrixKD\_CHnvNi  
0.0027 0.27  
\*\*  
loguniform  
MatrixKD\_CHnzNi  
0.0014 0.14  
\*\*  
loguniform  
MatrixKD\_PPw\_Ni  
0.0009 0.09  
\*\*  
loguniform  
MatrixKD\_UCF\_Ni  
0.0011 0.11  
\*\*  
loguniform  
MatrixKD\_BFw\_Ni  
0.001 0.1  
\*\*  
loguniform  
MatrixKD\_TR\_\_Ni  
0.001 0.1  
\*\*  
loguniform  
MatrixKD\_MCF\_Ni  
0.0011 0.11  
\*\*  
constant  
MatrixKD\_LAF\_C  
0.0  
\*\*  
constant  
MatrixKD\_UAF\_C  
0.0  
\*\*  
constant  
MatrixKD\_AV\_\_C  
0.0  
\*\*  
constant  
MatrixKD\_tac\_C  
0.0  
\*\*  
constant  
MatrixKD\_TCw\_C  
0.0  
\*\*  
constant

MatrixKD\_PTn\_C  
0.0  
\*\*  
constant  
MatrixKD\_TSw\_C  
0.0  
\*\*  
constant  
MatrixKD\_TSv\_C  
0.0  
\*\*  
constant  
MatrixKD\_CHnvC  
0.0  
\*\*  
constant  
MatrixKD\_CHnzC  
0.0  
\*\*  
constant  
MatrixKD\_PPw\_C  
0.0  
\*\*  
constant  
MatrixKD\_UCF\_C  
0.0  
\*\*  
constant  
MatrixKD\_BFw\_C  
0.0  
\*\*  
constant  
MatrixKD\_TR\_\_C  
0.0  
\*\*  
constant  
MatrixKD\_MCF\_C  
0.0  
\*\*  
constant  
MatrixKD\_LAF\_Se  
1.0  
\*\*  
constant  
MatrixKD\_UAF\_Se  
1.0  
\*\*  
constant  
MatrixKD\_AV\_\_Se  
1.0  
\*\*

```
constant
MatrixKD_tac_Se
1.0
**
constant
MatrixKD_TCw_Se
1.0
**
constant
MatrixKD_PTn_Se
1.0
**
loguniform
MatrixKD_TSw_Se
2.6e-4  0.026
**
loguniform
MatrixKD_TVv_Se
2.6e-4  0.026
**
loguniform
MatrixKD_CHnvSe
3.e-4   0.03
**
loguniform
MatrixKD_CHnzSe
4.5e-4  0.045
**
loguniform
MatrixKD_PPw_Se
0.00025 0.025
**
loguniform
MatrixKD_UCF_Se
0.00036 0.036
**
loguniform
MatrixKD_BFw_Se
0.0013  0.13
**
loguniform
MatrixKD_TR__Se
0.0013  0.13
**
loguniform
MatrixKD_MCF_Se
0.00036 0.036
**
constant
MatrixKD_LAF_Nb
0.0
```

```
**
constant
MatrixKD_UAF_Nb
0.0
**
constant
MatrixKD_AV_Nb
0.0
**
constant
MatrixKD_tac_Nb
0.0
**
constant
MatrixKD_TCw_Nb
0.0
**
constant
MatrixKD_PTn_Nb
0.0
**
constant
MatrixKD_TSw_Nb
0.0
**
constant
MatrixKD_Tsv_Nb
0.0
**
constant
MatrixKD_CHnvNb
0.0
**
constant
MatrixKD_CHnzNb
0.0
**
constant
MatrixKD_PPw_Nb
0.0
**
constant
MatrixKD_UCF_Nb
0.0
**
constant
MatrixKD_BFw_Nb
0.0
**
constant
MatrixKD_TR__Nb
```

0.0  
\*\*  
constant  
MatrixKD\_MCF\_Nb  
0.0  
\*\*  
constant  
FractureRD\_LAF\_Am  
1.0  
\*\*  
constant  
FractureRD\_UAF\_Am  
1.0  
\*\*  
constant  
FractureRD\_AV\_\_Am  
1.0  
\*\*  
constant  
FractureRD\_tac\_Am  
1.0  
\*\*  
constant  
FractureRD\_TCw\_Am  
1.0  
\*\*  
constant  
FractureRD\_PTn\_Am  
1.0  
\*\*  
constant  
FractureRD\_TSw\_Am  
1.0  
\*\*  
constant  
FractureRD\_Tsv\_Am  
1.0  
\*\*  
constant  
FractureRD\_CHnvAm  
1.0  
\*\*  
constant  
FractureRD\_CHnzAm  
1.0  
\*\*  
constant  
FractureRD\_PPw\_Am  
1.0  
\*\*  
constant

FractureRD\_UCF\_Am  
1.0  
\*\*  
constant  
FractureRD\_BFw\_Am  
1.0  
\*\*  
constant  
FractureRD\_TR\_\_Am  
1.0  
\*\*  
constant  
FractureRD\_MCF\_Am  
1.0  
\*\*  
constant  
FractureRD\_LAF\_Np  
1.0  
\*\*  
constant  
FractureRD\_UAF\_Np  
1.0  
\*\*  
constant  
FractureRD\_AV\_\_Np  
1.0  
\*\*  
constant  
FractureRD\_tac\_Np  
1.0  
\*\*  
constant  
FractureRD\_TCw\_Np  
1.0  
\*\*  
constant  
FractureRD\_PTn\_Np  
1.0  
\*\*  
constant  
FractureRD\_TSw\_Np  
1.0  
\*\*  
constant  
FractureRD\_TSv\_Np  
1.0  
\*\*  
constant  
FractureRD\_CHnvNp  
1.0  
\*\*

```
constant
FractureRD_CHnzNp
1.0
**
constant
FractureRD_PPw_Np
1.0
**
constant
FractureRD_UCF_Np
1.0
**
constant
FractureRD_BFw_Np
1.0
**
constant
FractureRD_TR__Np
1.0
**
constant
FractureRD_MCF_Np
1.0
**
constant
FractureRD_LAF_I
1.0
**
constant
FractureRD_UAF_I
1.0
**
constant
FractureRD_AV__I
1.0
**
constant
FractureRD_tac_I
1.0
**
constant
FractureRD_TCw_I
1.0
**
constant
FractureRD_PTn_I
1.0
**
constant
FractureRD_TSw_I
1.0
```

```
**
constant
FractureRD_TSv_I
1.0
**
constant
FractureRD_CHnvI
1.0
**
constant
FractureRD_CHnzI
1.0
**
constant
FractureRD_PPw_I
1.0
**
constant
FractureRD_UCF_I
1.0
**
constant
FractureRD_BFw_I
1.0
**
constant
FractureRD_TR_I
1.0
**
constant
FractureRD_MCF_I
1.0
**
constant
FractureRD_LAF_Tc
1.0
**
constant
FractureRD_UAF_Tc
1.0
**
constant
FractureRD_AV__Tc
1.0
**
constant
FractureRD_tac_Tc
1.0
**
constant
FractureRD_TCw_Tc
```

1.0  
\*\*  
constant  
FractureRD\_PTn\_Tc  
1.0  
\*\*  
constant  
FractureRD\_TSw\_Tc  
1.0  
\*\*  
constant  
FractureRD\_Tsv\_Tc  
1.0  
\*\*  
constant  
FractureRD\_CHnvTc  
1.0  
\*\*  
constant  
FractureRD\_CHnzTc  
1.0  
\*\*  
constant  
FractureRD\_PPw\_Tc  
1.0  
\*\*  
constant  
FractureRD\_UCF\_Tc  
1.0  
\*\*  
constant  
FractureRD\_BFw\_Tc  
1.0  
\*\*  
constant  
FractureRD\_TR\_\_Tc  
1.0  
\*\*  
constant  
FractureRD\_MCF\_Tc  
1.0  
\*\*  
constant  
FractureRD\_LAF\_CI  
1.0  
\*\*  
constant  
FractureRD\_UAF\_CI  
1.0  
\*\*  
constant

FractureRD\_AV\_CI

1.0

\*\*

constant

FractureRD\_tac\_CI

1.0

\*\*

constant

FractureRD\_TCw\_CI

1.0

\*\*

constant

FractureRD\_PTn\_CI

1.0

\*\*

constant

FractureRD\_TSw\_CI

1.0

\*\*

constant

FractureRD\_Tsv\_CI

1.0

\*\*

constant

FractureRD\_CHnvCI

1.0

\*\*

constant

FractureRD\_CHnzCI

1.0

\*\*

constant

FractureRD\_PPw\_CI

1.0

\*\*

constant

FractureRD\_UCF\_CI

1.0

\*\*

constant

FractureRD\_BFw\_CI

1.0

\*\*

constant

FractureRD\_TR\_CI

1.0

\*\*

constant

FractureRD\_MCF\_CI

1.0

\*\*

constant  
FractureRD\_LAF\_Cm  
1.0  
\*\*  
constant  
FractureRD\_UAF\_Cm  
1.0  
\*\*  
constant  
FractureRD\_AV\_\_Cm  
1.0  
\*\*  
constant  
FractureRD\_tac\_Cm  
1.0  
\*\*  
constant  
FractureRD\_TCw\_Cm  
1.0  
\*\*  
constant  
FractureRD\_PTn\_Cm  
1.0  
\*\*  
constant  
FractureRD\_TSw\_Cm  
1.0  
\*\*  
constant  
FractureRD\_Tsv\_Cm  
1.0  
\*\*  
constant  
FractureRD\_CHnvCm  
1.0  
\*\*  
constant  
FractureRD\_CHnzCm  
1.0  
\*\*  
constant  
FractureRD\_PPw\_Cm  
1.0  
\*\*  
constant  
FractureRD\_UCF\_Cm  
1.0  
\*\*  
constant  
FractureRD\_BFw\_Cm  
1.0

\*\*  
constant  
FractureRD\_TR\_\_Cm  
1.0  
\*\*  
constant  
FractureRD\_MCF\_Cm  
1.0  
\*\*  
constant  
FractureRD\_LAF\_U  
1.0  
\*\*  
constant  
FractureRD\_UAF\_U  
1.0  
\*\*  
constant  
FractureRD\_AV\_\_U  
1.0  
\*\*  
constant  
FractureRD\_tac\_U  
1.0  
\*\*  
constant  
FractureRD\_TCw\_U  
1.0  
\*\*  
constant  
FractureRD\_PTn\_U  
1.0  
\*\*  
constant  
FractureRD\_TSw\_U  
1.0  
\*\*  
constant  
FractureRD\_TSv\_U  
1.0  
\*\*  
constant  
FractureRD\_CHnvU  
1.0  
\*\*  
constant  
FractureRD\_CHnzU  
1.0  
\*\*  
constant  
FractureRD\_PPw\_U

1.0  
\*\*  
constant  
FractureRD\_UCF\_U  
1.0  
\*\*  
constant  
FractureRD\_BFw\_U  
1.0  
\*\*  
constant  
FractureRD\_TR\_\_U  
1.0  
\*\*  
constant  
FractureRD\_MCF\_U  
1.0  
\*\*  
constant  
FractureRD\_LAF\_Pu  
1.0  
\*\*  
constant  
FractureRD\_UAF\_Pu  
1.0  
\*\*  
constant  
FractureRD\_AV\_\_Pu  
1.0  
\*\*  
constant  
FractureRD\_tac\_Pu  
1.0  
\*\*  
constant  
FractureRD\_TCw\_Pu  
1.0  
\*\*  
constant  
FractureRD\_PTn\_Pu  
1.0  
\*\*  
constant  
FractureRD\_TSw\_Pu  
1.0  
\*\*  
constant  
FractureRD\_Tsv\_Pu  
1.0  
\*\*  
constant

FractureRD\_CHnvPu  
1.0  
\*\*  
constant  
FractureRD\_CHnzPu  
1.0  
\*\*  
constant  
FractureRD\_PPw\_Pu  
1.0  
\*\*  
constant  
FractureRD\_UCF\_Pu  
1.0  
\*\*  
constant  
FractureRD\_BFw\_Pu  
1.0  
\*\*  
constant  
FractureRD\_TR\_\_Pu  
1.0  
\*\*  
constant  
FractureRD\_MCF\_Pu  
1.0  
\*\*  
constant  
FractureRD\_LAF\_Th  
1.0  
\*\*  
constant  
FractureRD\_UAF\_Th  
1.0  
\*\*  
constant  
FractureRD\_AV\_\_Th  
1.0  
\*\*  
constant  
FractureRD\_tac\_Th  
1.0  
\*\*  
constant  
FractureRD\_TCw\_Th  
1.0  
\*\*  
constant  
FractureRD\_PTn\_Th  
1.0  
\*\*

```
constant
FractureRD_TSw_Th
1.0
**
constant
FractureRD_TSV_Th
1.0
**
constant
FractureRD_CHnvTh
1.0
**
constant
FractureRD_CHnzTh
1.0
**
constant
FractureRD_PPw_Th
1.0
**
constant
FractureRD_UCF_Th
1.0
**
constant
FractureRD_BFw_Th
1.0
**
constant
FractureRD_TR__Th
1.0
**
constant
FractureRD_MCF_Th
1.0
**
constant
FractureRD_LAF_Ra
1.0
**
constant
FractureRD_UAF_Ra
1.0
**
constant
FractureRD_AV__Ra
1.0
**
constant
FractureRD_tac_Ra
1.0
```

```
**
constant
FractureRD_TCw_Ra
1.0
**
constant
FractureRD_PTn_Ra
1.0
**
constant
FractureRD_TSw_Ra
1.0
**
constant
FractureRD_TSv_Ra
1.0
**
constant
FractureRD_CHnvRa
1.0
**
constant
FractureRD_CHnzRa
1.0
**
constant
FractureRD_PPw_Ra
1.0
**
constant
FractureRD_UCF_Ra
1.0
**
constant
FractureRD_BFw_Ra
1.0
**
constant
FractureRD_TR__Ra
1.0
**
constant
FractureRD_MCF_Ra
1.0
**
constant
FractureRD_LAF_Pb
1.0
**
constant
FractureRD_UAF_Pb
```

1.0  
\*\*  
constant  
FractureRD\_AV\_\_Pb  
1.0  
\*\*  
constant  
FractureRD\_tac\_Pb  
1.0  
\*\*  
constant  
FractureRD\_TCw\_Pb  
1.0  
\*\*  
constant  
FractureRD\_PTn\_Pb  
1.0  
\*\*  
constant  
FractureRD\_TSw\_Pb  
1.0  
\*\*  
constant  
FractureRD\_Tsv\_Pb  
1.0  
\*\*  
constant  
FractureRD\_CHnvPb  
1.0  
\*\*  
constant  
FractureRD\_CHnzPb  
1.0  
\*\*  
constant  
FractureRD\_PPw\_Pb  
1.0  
\*\*  
constant  
FractureRD\_UCF\_Pb  
1.0  
\*\*  
constant  
FractureRD\_BFw\_Pb  
1.0  
\*\*  
constant  
FractureRD\_TR\_\_Pb  
1.0  
\*\*  
constant

FractureRD\_MCF\_Pb  
1.0  
\*\*  
constant  
FractureRD\_LAF\_Cs  
1.0  
\*\*  
constant  
FractureRD\_UAF\_Cs  
1.0  
\*\*  
constant  
FractureRD\_AV\_Cs  
1.0  
\*\*  
constant  
FractureRD\_tac\_Cs  
1.0  
\*\*  
constant  
FractureRD\_TCw\_Cs  
1.0  
\*\*  
constant  
FractureRD\_PTn\_Cs  
1.0  
\*\*  
constant  
FractureRD\_TSw\_Cs  
1.0  
\*\*  
constant  
FractureRD\_TSv\_Cs  
1.0  
\*\*  
constant  
FractureRD\_CHnvCs  
1.0  
\*\*  
constant  
FractureRD\_CHnzCs  
1.0  
\*\*  
constant  
FractureRD\_PPw\_Cs  
1.0  
\*\*  
constant  
FractureRD\_UCF\_Cs  
1.0  
\*\*

```
constant
FractureRD_BFw_Cs
1.0
**
constant
FractureRD_TR__Cs
1.0
**
constant
FractureRD_MCF_Cs
1.0
**
constant
FractureRD_LAF_Ni
1.0
**
constant
FractureRD_UAF_Ni
1.0
**
constant
FractureRD_AV__Ni
1.0
**
constant
FractureRD_tac_Ni
1.0
**
constant
FractureRD_TCw_Ni
1.0
**
constant
FractureRD_PTn_Ni
1.0
**
constant
FractureRD_TSw_Ni
1.0
**
constant
FractureRD_TSv_Ni
1.0
**
constant
FractureRD_CHnvNi
1.0
**
constant
FractureRD_CHnzNi
1.0
```

\*\*  
constant  
FractureRD\_PPw\_Ni  
1.0  
\*\*  
constant  
FractureRD\_UCF\_Ni  
1.0  
\*\*  
constant  
FractureRD\_BFw\_Ni  
1.0  
\*\*  
constant  
FractureRD\_TR\_Ni  
1.0  
\*\*  
constant  
FractureRD\_MCF\_Ni  
1.0  
\*\*  
constant  
FractureRD\_LAF\_C  
1.0  
\*\*  
constant  
FractureRD\_UAF\_C  
1.0  
\*\*  
constant  
FractureRD\_AV\_C  
1.0  
\*\*  
constant  
FractureRD\_tac\_C  
1.0  
\*\*  
constant  
FractureRD\_TCw\_C  
1.0  
\*\*  
constant  
FractureRD\_PTn\_C  
1.0  
\*\*  
constant  
FractureRD\_TSw\_C  
1.0  
\*\*  
constant  
FractureRD\_Tsv\_C

```
1.0
**
constant
FractureRD_CHnvC
1.0
**
constant
FractureRD_CHnzC
1.0
**
constant
FractureRD_PPw_C
1.0
**
constant
FractureRD_UCF_C
1.0
**
constant
FractureRD_BFw_C
1.0
**
constant
FractureRD_TR__C
1.0
**
constant
FractureRD_MCF_C
1.0
**
constant
FractureRD_LAF_Se
1.0
**
constant
FractureRD_UAF_Se
1.0
**
constant
FractureRD_AV__Se
1.0
**
constant
FractureRD_tac_Se
1.0
**
constant
FractureRD_TCw_Se
1.0
**
constant
```

FractureRD\_PTn\_Se  
1.0  
\*\*  
constant  
FractureRD\_TSw\_Se  
1.0  
\*\*  
constant  
FractureRD\_TSv\_Se  
1.0  
\*\*  
constant  
FractureRD\_CHnvSe  
1.0  
\*\*  
constant  
FractureRD\_CHnzSe  
1.0  
\*\*  
constant  
FractureRD\_PPw\_Se  
1.0  
\*\*  
constant  
FractureRD\_UCF\_Se  
1.0  
\*\*  
constant  
FractureRD\_BFw\_Se  
1.0  
\*\*  
constant  
FractureRD\_TR\_\_Se  
1.0  
\*\*  
constant  
FractureRD\_MCF\_Se  
1.0  
\*\*  
constant  
FractureRD\_LAF\_Nb  
1.0  
\*\*  
constant  
FractureRD\_UAF\_Nb  
1.0  
\*\*  
constant  
FractureRD\_AV\_\_Nb  
1.0  
\*\*

```
constant
FractureRD_tac_Nb
1.0
**
constant
FractureRD_TCw_Nb
1.0
**
constant
FractureRD_PTn_Nb
1.0
**
constant
FractureRD_TSw_Nb
1.0
**
constant
FractureRD_Tsv_Nb
1.0
**
constant
FractureRD_CHnvNb
1.0
**
constant
FractureRD_CHnzNb
1.0
**
constant
FractureRD_PPw_Nb
1.0
**
constant
FractureRD_UCF_Nb
1.0
**
constant
FractureRD_BFw_Nb
1.0
**
constant
FractureRD_TR__Nb
1.0
**
constant
FractureRD_MCF_Nb
1.0
**
constant
MatrixPermeability_LAF_[m2]
0.0
```

```
**
constant
MatrixPermeability_UAF_[m2]
0.0
**
constant
MatrixPermeability_AV__[m2]
0.0
**
constant
MatrixPermeability_tac_[m2]
0.0
**
constant
MatrixPermeability_TCw_[m2]
0.0
**
constant
MatrixPermeability_PTn_[m2]
0.0
**
lognormal
MatrixPermeability_TSw_[m2]
2.6e-20 7.3e-17
**
lognormal
MatrixPermeability_TSv_[m2]
2.6e-20 7.3e-17
**
lognormal
MatrixPermeability_CHnv[m2]
7.2e-19 1.6e-14
**
lognormal
MatrixPermeability_CHnz[m2]
1.3e-20 5.8e-17
**
lognormal
MatrixPermeability_PPw_[m2]
1.0e-18 2.3e-14
**
lognormal
MatrixPermeability_UCF_[m2]
1.8e-17 4.4e-15
**
lognormal
MatrixPermeability_BFw_[m2]
1.8e-18 2.1e-16
**
lognormal
MatrixPermeability_TR__[m2]
```

```
1.8e-18 2.1e-16
**
lognormal
MatrixPermeability_MCF_[m2]
1.7e-17 4.4e-15
**
constant
MatrixPorosity_LAF_
0.20
**
constant
MatrixPorosity_UAF_
0.20
**
constant
MatrixPorosity_AV__
0.20
**
constant
MatrixPorosity_tac_
0.20
**
constant
MatrixPorosity_TCw_
0.20
**
constant
MatrixPorosity_PTn_
0.40
**
uniform
MatrixPorosity_TSw_
0.05 0.22
**
uniform
MatrixPorosity_Tsv_
0.05 0.22
**
uniform
MatrixPorosity_CHnv
0.18 0.49
**
uniform
MatrixPorosity_CHnz
0.22 0.39
**
uniform
MatrixPorosity_PPw_
0.15 0.44
**
uniform
```

MatrixPorosity\_UCF\_

0.15 0.37

\*\*

uniform

MatrixPorosity\_BFw\_

0.08 0.25

\*\*

uniform

MatrixPorosity\_TR\_\_

0.08 0.25

\*\*

uniform

MatrixPorosity\_MCF\_

0.15 0.37

\*\*

constant

MatrixBeta\_LAF\_

0.0

\*\*

constant

MatrixBeta\_UAF\_

0.0

\*\*

constant

MatrixBeta\_AV\_\_

0.0

\*\*

constant

MatrixBeta\_tac\_

0.0

\*\*

constant

MatrixBeta\_TCw\_

1.607

\*\*

constant

MatrixBeta\_PTn\_

2.223

\*\*

uniform

MatrixBeta\_TSw\_

1.16 2.71

\*\*

uniform

MatrixBeta\_TSv\_

1.16 2.71

\*\*

uniform

MatrixBeta\_CHnv

1.05 7.05

\*\*

uniform  
MatrixBeta\_CHnz  
1.19 2.53  
\*\*

uniform  
MatrixBeta\_PPw\_  
1.92 2.65  
\*\*

uniform  
MatrixBeta\_UCF\_  
0.979 3.25  
\*\*

uniform  
MatrixBeta\_BFw\_  
1.87 5.53  
\*\*

uniform  
MatrixBeta\_TR\_\_  
1.87 5.53  
\*\*

uniform  
MatrixBeta\_MCF\_  
0.978 3.23  
\*\*

constant  
MatrixGrainDensity\_LAF\_[kg/m3]  
2300.  
\*\*

constant  
MatrixGrainDensity\_UAF\_[kg/m3]  
2300.  
\*\*

constant  
MatrixGrainDensity\_AV\_\_[kg/m3]  
2300.  
\*\*

constant  
MatrixGrainDensity\_tac\_[kg/m3]  
2300.  
\*\*

constant  
MatrixGrainDensity\_TCw\_[kg/m3]  
2300.  
\*\*

constant  
MatrixGrainDensity\_PTn\_[kg/m3]  
1400.  
\*\*

constant  
MatrixGrainDensity\_TSw\_[kg/m3]  
2300.0

\*\*  
constant  
MatrixGrainDensity\_TSv\_[kg/m3]  
2580.  
\*\*  
constant  
MatrixGrainDensity\_CHnv[kg/m3]  
2300.0  
\*\*  
constant  
MatrixGrainDensity\_CHnz[kg/m3]  
2300.0  
\*\*  
constant  
MatrixGrainDensity\_PPw\_[kg/m3]  
2590.0  
\*\*  
constant  
MatrixGrainDensity\_UCF\_[kg/m3]  
2270.0  
\*\*  
constant  
MatrixGrainDensity\_BFw\_[kg/m3]  
2630.0  
\*\*  
constant  
MatrixGrainDensity\_TR\_\_[kg/m3]  
2630.  
\*\*  
constant  
MatrixGrainDensity\_MCF\_[kg/m3]  
2630.0  
\*\*  
constant  
FracturePermeability\_LAF\_[m2]  
0.0  
\*\*  
constant  
FracturePermeability\_UAF\_[m2]  
0.0  
\*\*  
constant  
FracturePermeability\_AV\_\_[m2]  
0.0  
\*\*  
constant  
FracturePermeability\_tac\_[m2]  
0.0  
\*\*  
constant  
FracturePermeability\_TCw\_[m2]

1.0e-16  
\*\*  
constant  
FracturePermeability\_PTn\_[m2]  
1.0e-15  
\*\*  
lognormal  
FracturePermeability\_TSw\_[m2]  
1.0e-12 1.0e-11  
\*\*  
lognormal  
FracturePermeability\_TSv\_[m2]  
1.0e-12 1.0e-11  
\*\*  
lognormal  
FracturePermeability\_CHnv[m2]  
1.0e-13 1.0e-11  
\*\*  
lognormal  
FracturePermeability\_CHnz[m2]  
1.0e-13 1.0e-11  
\*\*  
lognormal  
FracturePermeability\_PPw\_[m2]  
1.0e-13 1.0e-11  
\*\*  
lognormal  
FracturePermeability\_UCF\_[m2]  
1.0e-13 1.0e-11  
\*\*  
lognormal  
FracturePermeability\_BFw\_[m2]  
1.0e-13 1.0e-11  
\*\*  
lognormal  
FracturePermeability\_TR\_\_[m2]  
1.0e-13 1.0e-11  
\*\*  
lognormal  
FracturePermeability\_MCF\_[m2]  
1.0e-13 1.0e-11  
\*\*  
lognormal  
FracturePorosity\_LAF\_  
3.0e-3 7.0e-2  
\*\*  
lognormal  
FracturePorosity\_UAF\_  
3.0e-3 7.0e-2  
\*\*  
lognormal

FracturePorosity\_AV\_\_

3.0e-3 7.0e-2

\*\*

lognormal

FracturePorosity\_tac\_

3.0e-3 7.0e-2

\*\*

lognormal

FracturePorosity\_TCw\_

3.0e-3 7.0e-2

\*\*

lognormal

FracturePorosity\_PTn\_

3.0e-3 7.0e-2

\*\*

lognormal

FracturePorosity\_TSw\_

3.0e-3 7.0e-2

\*\*

lognormal

FracturePorosity\_TSw\_

3.0e-3 7.0e-2

\*\*

lognormal

FracturePorosity\_CHnv

3.0e-3 7.0e-2

\*\*

lognormal

FracturePorosity\_CHnz

3.0e-3 7.0e-2

\*\*

lognormal

FracturePorosity\_PPw\_

3.0e-3 7.0e-2

\*\*

lognormal

FracturePorosity\_UCF\_

3.0e-3 7.0e-2

\*\*

lognormal

FracturePorosity\_BFw\_

3.0e-3 7.0e-2

\*\*

lognormal

FracturePorosity\_TR\_\_

3.0e-3 7.0e-2

\*\*

lognormal

FracturePorosity\_MCF\_

3.0e-3 7.0e-2

\*\*

```
constant
FractureBeta_LAF_
0.0
**
constant
FractureBeta_UAF_
0.0
**
constant
FractureBeta_AV__
0.0
**
constant
FractureBeta_tac_
0.0
**
constant
FractureBeta_TCw_
0.0
**
constant
FractureBeta_PTn_
0.0
**
uniform
FractureBeta_TSw_
1.45 12.3
**
uniform
FractureBeta_Tsv_
1.45 12.3
**
uniform
FractureBeta_CHnv
1.45 12.3
**
uniform
FractureBeta_CHnz
1.45 12.3
**
uniform
FractureBeta_PPw_
1.45 12.3
**
uniform
FractureBeta_UCF_
1.45 12.3
**
uniform
FractureBeta_BFw_
1.45 12.3
```

```
**
uniform
FractureBeta_TR__
1.45 12.3
**
uniform
FractureBeta_MCF_
1.45 12.3
**
constant
InletArea__1SubArea[m2]
5.4e5
**
constant
InletArea__2SubArea[m2]
5.4e5
**
constant
InletArea__3SubArea[m2]
5.4e5
**
constant
InletArea__4SubArea[m2]
5.4e5
**
constant
InletArea__5SubArea[m2]
5.4e5
**
constant
InletArea__6SubArea[m2]
5.4e5
**
constant
OutletArea__1SubArea[m2]
2.62e5
**
constant
OutletArea__2SubArea[m2]
2.62e5
**
constant
OutletArea__3SubArea[m2]
2.62e5
**
constant
OutletArea__4SubArea[m2]
2.62e5
**
constant
OutletArea__5SubArea[m2]
```

```
2.62e5
**
constant
OutletArea__6SubArea[m2]
2.62e5
**
constant
LAF_Thickness_1SubArea[m]
0.0
**
constant
UAF_Thickness_1SubArea[m]
0.0
**
constant
AV__Thickness_1SubArea[m]
0.0
**
constant
tac_Thickness_1SubArea[m]
0.0
**
constant
TCw_Thickness_1SubArea[m]
0.0
**
constant
PTn_Thickness_1SubArea[m]
0.0
**
constant
TSw_Thickness_1SubArea[m]
105.0
**
constant
TSv_Thickness_1SubArea[m]
8.0
**
constant
CHnvThickness_1SubArea[m]
90.0
**
constant
CHnzThickness_1SubArea[m]
60.0
**
constant
PPw_Thickness_1SubArea[m]
90.0
**
constant
```

UCF\_Thickness\_1SubArea[m]  
0.0  
\*\*  
constant  
BFw\_Thickness\_1SubArea[m]  
0.0  
\*\*  
constant  
TR\_\_Thickness\_1SubArea[m]  
0.0  
\*\*  
constant  
MCF\_Thickness\_1SubArea[m]  
0.0  
\*\*  
constant  
LAF\_Thickness\_2SubArea[m]  
0.0  
\*\*  
constant  
UAF\_Thickness\_2SubArea[m]  
0.0  
\*\*  
constant  
AV\_\_Thickness\_2SubArea[m]  
0.0  
\*\*  
constant  
tac\_Thickness\_2SubArea[m]  
0.0  
\*\*  
constant  
TCw\_Thickness\_2SubArea[m]  
0.0  
\*\*  
constant  
PTn\_Thickness\_2SubArea[m]  
0.0  
\*\*  
constant  
TSw\_Thickness\_2SubArea[m]  
116.0  
\*\*  
constant  
TSv\_Thickness\_2SubArea[m]  
0.0  
\*\*  
constant  
CHnvThickness\_2SubArea[m]  
51.0  
\*\*

```
constant
CHnzThickness_2SubArea[m]
138.0
**
constant
PPw_Thickness_2SubArea[m]
0.0
**
constant
UCF_Thickness_2SubArea[m]
0.0
**
constant
BFw_Thickness_2SubArea[m]
0.0
**
constant
TR__Thickness_2SubArea[m]
0.0
**
constant
MCF_Thickness_2SubArea[m]
0.0
**
constant
LAF_Thickness_3SubArea[m]
0.0
**
constant
UAF_Thickness_3SubArea[m]
0.0
**
constant
AV__Thickness_3SubArea[m]
0.0
**
constant
tac_Thickness_3SubArea[m]
0.0
**
constant
TCw_Thickness_3SubArea[m]
0.0
**
constant
PTn_Thickness_3SubArea[m]
0.0
**
constant
TSw_Thickness_3SubArea[m]
119.0
```

```
**
constant
TSv_Thickness_3SubArea[m]
0.0
**
constant
CHnvThickness_3SubArea[m]
56.0
**
constant
CHnzThickness_3SubArea[m]
146.0
**
constant
PPw_Thickness_3SubArea[m]
0.0
**
constant
UCF_Thickness_3SubArea[m]
0.0
**
constant
BFw_Thickness_3SubArea[m]
0.0
**
constant
TR__Thickness_3SubArea[m]
0.0
**
constant
MCF_Thickness_3SubArea[m]
0.0
**
constant
LAF_Thickness_4SubArea[m]
0.0
**
constant
UAF_Thickness_4SubArea[m]
0.0
**
constant
AV__Thickness_4SubArea[m]
0.0
**
constant
tac_Thickness_4SubArea[m]
0.0
**
constant
TCw_Thickness_4SubArea[m]
```

0.0  
\*\*  
constant  
PTn\_Thickness\_4SubArea[m]  
0.0  
\*\*  
constant  
TSw\_Thickness\_4SubArea[m]  
81.0  
\*\*  
constant  
TSv\_Thickness\_4SubArea[m]  
0.0  
\*\*  
constant  
CHnvThickness\_4SubArea[m]  
84.0  
\*\*  
constant  
CHnzThickness\_4SubArea[m]  
150.0  
\*\*  
constant  
PPw\_Thickness\_4SubArea[m]  
0.0  
\*\*  
constant  
UCF\_Thickness\_4SubArea[m]  
0.0  
\*\*  
constant  
BFw\_Thickness\_4SubArea[m]  
0.0  
\*\*  
constant  
TR\_\_Thickness\_4SubArea[m]  
0.0  
\*\*  
constant  
MCF\_Thickness\_4SubArea[m]  
0.0  
\*\*  
constant  
LAF\_Thickness\_5SubArea[m]  
0.0  
\*\*  
constant  
UAF\_Thickness\_5SubArea[m]  
0.0  
\*\*  
constant

AV\_Thickness\_5SubArea[m]  
0.0  
\*\*  
constant  
tac\_Thickness\_5SubArea[m]  
0.0  
\*\*  
constant  
TCw\_Thickness\_5SubArea[m]  
0.0  
\*\*  
constant  
PTn\_Thickness\_5SubArea[m]  
0.0  
\*\*  
constant  
TSw\_Thickness\_5SubArea[m]  
124.0  
\*\*  
constant  
TSv\_Thickness\_5SubArea[m]  
0.0  
\*\*  
constant  
CHnvThickness\_5SubArea[m]  
78.0  
\*\*  
constant  
CHnzThickness\_5SubArea[m]  
128.0  
\*\*  
constant  
PPw\_Thickness\_5SubArea[m]  
0.0  
\*\*  
constant  
UCF\_Thickness\_5SubArea[m]  
0.0  
\*\*  
constant  
BFw\_Thickness\_5SubArea[m]  
0.0  
\*\*  
constant  
TR\_Thickness\_5SubArea[m]  
0.0  
\*\*  
constant  
MCF\_Thickness\_5SubArea[m]  
0.0  
\*\*

```
constant
LAF_Thickness_6SubArea[m]
0.0
**
constant
UAF_Thickness_6SubArea[m]
0.0
**
constant
AV__Thickness_6SubArea[m]
0.0
**
constant
tac_Thickness_6SubArea[m]
0.0
**
constant
TCw_Thickness_6SubArea[m]
0.0
**
constant
PTn_Thickness_6SubArea[m]
0.0
**
constant
TSw_Thickness_6SubArea[m]
54.0
**
constant
TSv_Thickness_6SubArea[m]
0.0
**
constant
CHnvThickness_6SubArea[m]
79.0
**
constant
CHnzThickness_6SubArea[m]
191.0
**
constant
PPw_Thickness_6SubArea[m]
0.0
**
constant
UCF_Thickness_6SubArea[m]
0.0
**
constant
BFw_Thickness_6SubArea[m]
0.0
```

```
**
constant
TR_Thickness_6SubArea[m]
0.0
**
constant
MCF_Thickness_6SubArea[m]
0.0
**
constant
SaturatedZoneSourceLength_1SubArea[m]
150.
**
constant
SaturatedZoneThickness_LAF__1SubArea[m]
0.0
**
constant
SaturatedZoneThickness_UAF__1SubArea[m]
0.0
**
constant
SaturatedZoneThickness_AV___1SubArea[m]
0.0
**
constant
SaturatedZoneThickness_tac__1SubArea[m]
0.0
**
constant
SaturatedZoneThickness_TCw__1SubArea[m]
0.0
**
constant
SaturatedZoneThickness_PTn__1SubArea[m]
0.0
**
constant
SaturatedZoneThickness_TSw__1SubArea[m]
0.0
**
constant
SaturatedZoneThickness_TSw__1SubArea[m]
0.0
**
constant
SaturatedZoneThickness_TSv__1SubArea[m]
0.0
**
constant
SaturatedZoneThickness_CHnv_1SubArea[m]
0.0
**
constant
SaturatedZoneThickness_CHnz_1SubArea[m]
```

```
100.0
**
constant
SaturatedZoneThickness_PPw__1SubArea[m]
400.0
**
constant
SaturatedZoneThickness_UCF__1SubArea[m]
0.0
**
constant
SaturatedZoneThickness_BFw__1SubArea[m]
2750.0
**
constant
SaturatedZoneThickness_TR___1SubArea[m]
0.0
**
constant
SaturatedZoneThickness_MCF__1SubArea[m]
1950.0
**
ENDOFFILE
```

## **APPENDIX C**

# **DOCUMENTATION OF FUNCTIONS AND SUBROUTINES IN THE UTILITY MODULES**

# Documentation of Functions and Subroutines in the Utility Modules

## C.1 READER utility module (file=reader.f)

---

```
      subroutine reader( iunit, title1, title2,  
        & mxntime, ntim, tim,  
        & mxnucl, nnucl, names )
```

---

```
c used to read data from ".inp" file  
c called directly by EXEC  
c by Randall D. Manteufel, January 27, 1997  
c  
c iunit = input, integer, unit number that has file "tpa.inp" opened  
c           and ready to read from  
c title1 = output, character*80, first title line in tpa.inp file  
c title2 = output, character*80, second title line in tpa.inp file  
c mxntime = input, integer, dimension for time array  
c ntim    = output, integer, actual number of times used in time array  
c tim(mxntime) = output, double precision, array of times  
c mxnucl = input, integer, maximum dimension for nuclide array  
c nnucl  = output, integer, actual number of nuclides used  
c names(mxnucl)= output, character*6, names of nuclides used  
c  
c this will INITIALIZE the following:  
c   sampled parameter database  
c   module variable database  
c   subarea database  
c   invent database
```

---

```
      subroutine skipcommentline( iunit )
```

---

```
c read input file to skip comment lines that start with '*'  
c  
c iunit = input, integer, file unit number to read from  
c
```

---

```
      subroutine readname( iunit, name )
```

---

```
c read name of parameter from input file  
c  
c iunit = input, integer, file unit number to read from  
c name = output, character*60, name  
c
```

## C.2 FILE UNIT utility module (file=fileunit.f)

```
=====
integer function igetunitnumber( name )
=====
c get an open unit number to open file
c by Randall D. Manteufel, January 27, 1997
c
c name      = input, character*8, name of subroutine that wants
c            to open a file, hence needs a unit number. The
c            name must be recognized by TPA, hence must be one
c            of the following:
c            { uzflow, nfenv, ebsfail, ebsrel, uzft, szft,
c              dcagw, climato, seismo, volcano, faulto, ashplume,
c              ashremove, dcags, exec, test, utility }
c igetunitnumber = output, integer, open unit number that can be used
c                this unit number will be assigned only once
c                If one requests more than "maxunit" numbers then
c                an error will be printed and program stopped.
c
=====
subroutine printfun( )
=====
c print names of file unit numbers currently being used,
c this will be printed to standard output, 6
c by Randall D. Manteufel, February 4, 1997
c
```

### C.3 ARRAY utility module (file=array.f)

```
c array utility module
c by Randall D. Manteufel, January 27, 1997
c
c-----
      subroutine zero( n, v )
c-----
c zero-out vector
c
c n   = input, integer, length of vectors
c v[n] = output, double precision, vector to have all values set =0.0d0
c
c-----
      subroutine zeroi( n, iv )
c-----
c zero-out integer vector
c
c n   = input, integer, length of vectors
c iv[n] = output, integer, vector to have all values set =0
c
c-----
      subroutine clearchar( n, name )
c-----
c clear character array
c
c n = input, integer, dimension of array
c name = input/output, character*(n), array to be cleared, set = ' '
c
c-----
      subroutine transpose(i1, i2, n1, n2, a )
c-----
c transpose matrix
c
c i1, i2 = input, integer, original dimensions of matrix
c n1, n2 = input, integer, actual dimensions of matrix
c a[i1,i2] = input/output, double precision, matrix that
c           on output is transposed
c           Note: only transposes a(n1,n2) portion of matrix
c
c-----
      subroutine scale(c, n, v )
c-----
c scale vector
c
c c   = input, double precision, scaling factor
c n   = input, integer, length of vectors
c v[n] = input/output, double precision, first vector
c           v(new) = c* v(original)
c
c-----
```

```
subroutine scopy(c, n, vin, vout )
```

```
-----  
c scale and copy vector  
c  
c c      = input, double precision, scaling factor  
c n      = input, integer, length of vectors  
c vin[n] = input, double precision, first vector  
c vout[n] = output, double precision, output vector  
c          vout = c* vin  
c
```

```
-----  
subroutine acopy(c, n, vin, vout )
```

```
-----  
c copy vector and at same time add a constant to each entry  
c  
c c      = input, double precision, factor to be added to each entry  
c          of vin  
c n      = input, integer, length of vectors  
c vin(n) = input, double precision, first vector  
c vout(n) = output, double precision, output vector  
c          vout(i) = c + vin(i)  
c
```

```
-----  
subroutine ascopy(c, n, vin1, vin2, vout )
```

```
-----  
c add, scale and copy vector  
c  
c c      = input, double precision, scaling factor  
c n      = input, integer, length of vectors  
c vin1[n] = input, double precision, first vector  
c vin2[n] = input, double precision, second vector  
c vout[n] = output, double precision, output vector  
c          vout = c*( vin1 + vin2 )  
c
```

```
-----  
subroutine addto(n, vin1, vin2 )
```

```
-----  
c add one vector to another  
c  
c n      = input, integer, length of vectors  
c vin1[n] = input, double precision, first input vector  
c vin2[n] = input/output, double precision, second vector  
c          vin2 = vin1 + vin2  
c
```

```
-----  
subroutine isoneofset( iquery, nset, iset, index )
```

```
-----  
c determines if iquery is part of set iset, if so return index  
c for example, if nset = 3, iset = { 5, 99, 301 } and iquery = 99,  
c          then index = 2  
c for example, if nset = 5, iset = { 1, 2, 8, 9, 11 } and iquery = 3,
```

```

c         then index = 0
c
c iquery = input, integer, integer value to be checked to see if in
c         the integer set
c nset = input, integer, number of elements in the set
c iset(nset) = input, integer, set of integer values
c index = output, integer, index, or location, of iquery in the set,
c         if iquery is not found in set, then index = 0.
c         otherwise, 1 <= index <= nset
c

```

---

```

function pmean( n, x, m, y )

```

---

```

c calculate "significance probability" (also known as p-value)
c to determine if mean(x) = mean(y)
c the smaller the p-value the stronger the indication that
c the means are truly different.
c
c Method outlined in Mathematical Statistics and Data Analysis,
c by J.A. Rice, 1988 pg 356, Wadsworth&Brooks/Cole publishers.
c this is for twosided test
c if pmean<0.05 then normally reject null hypothesis
c and claim difference truly exists
c
c n     = input, integer, length of x-array values
c x[n]  = input, double precision, array of values
c m     = input, integer, length of y-array values
c y[m]  = input, double precision, array of values
c pmean = output, double precision, p-value
c       probability that mean(x) = mean(y)
c       Note:  0 < pmean < 1
c

```

---

```

function ainterl( n, t, v, tin )

```

---

```

c LINEAR interpolation in list of {time, value} data
c to find value at given time.
c
c n     = input, integer, length of array values
c t[n]  = input, double precision, array of times,
c       assumed to be in ascending order
c v[n]  = input, double precision, array of values
c tin   = input, double precision, time at which interpolated
c       value of "v" is requested
c ainterl = output, double precision, interpolated value
c       if tin < t(1), then ainterl = v(1)
c       if tin > t(n), then ainterl = v(n)
c

```

---

```

function avar( n, x )

```

---

```
c calculate variance of array of values
c
c n    = input, integer length of array values
c x[n] = input, double precision, array of values
c avar = output, double precision, variance of array of values
c
```

---

```
function amean( n, x )
```

---

```
c determine mean of array of values
c
c n    = input, integer length of array values
c x[n] = input, double precision, array of values
c amean = output, double precision, mean of array of values
c
```

---

```
subroutine checkinorder( n, x, icheck )
```

---

```
c determine if array of values are in order, either
c ascending or descending
c
c n    = input, integer length of array values
c x(n) = input, double precision, array of values
c icheck = output, integer,
c       = -1 if in descending order
c       = 0 if NOT in order
c       = +1 if in ascending order
c
```

---

```
subroutine checkforduplicates( n, x, icheck )
```

---

```
c determine if array of values has any duplicates
c assumes array already in ascending or descending order
c
c n    = input, integer length of array values
c x(n) = input, double precision, array of values
c icheck = output, integer,
c       = 0 if no duplicates found
c       = +1 if duplicates found
c
```

---

```
subroutine icheckforduplicates( n, ix, icheck )
```

---

```
c determine if array of values has any duplicates
c assumes array already in ascending or descending order
c
c n    = input, integer length of array values
c ix(n) = input, integer, array of values
c icheck = output, integer,
c       = 0 if no duplicates found
c       = +1 if duplicates found
```

```

c
c-----
c      subroutine sortqr(n, v, ipt)
c-----
c QUICK sort based on pointers to array of values
c sorts from smallest to largest
c
c n      = input, integer, number of elements to sort
c v[n]   = input, double precision, array of values to be sorted
c ipt[n] = input/output, integer, array of pointers
c        On INPUT, in most cases, ipt[1]=1, ipt[2]=2, etc. ipt[n]=n
c        each pointer must point to one unique v, so that
c        ipt[i] never equals ipt[j] except when i=j
c        also 1 <= ipt[i] <= n for all i
c        On OUTPUT, v[ipt[1]] is smallest, and
c            v[ipt[n]] is largest
c

```

```

c-----
c      subroutine maplist(n1, x1, y1, n2, x2, y2)
c-----
c map data in first list into second list
c given the first set of {x,y}, and the second set of {x}, find the
c second set of {y}
c
c n1 = input, integer, dimension of first array lists
c x1[n1] = input, double precision, array of x-values in first list
c y1[n1] = input, double precision, array of y-values in first list
c n2 = input, integer, dimension of second array lists
c x2[n2] = input, double precision, array of x-values in second list
c y2[n2] = output, double precision, array of y-values in second list
c        values of y2 based on linear interpolation to of first list
c

```

#### C.4 SAMPLER utility module (file=sampler.f)

c SAMPLER utility module

c by Randall D. Manteufel, January 27, 1997

---

subroutine newspdb()

---

c initialize new database for sampled parameters

c this is called only once, and by "reader"

c user does not call this subroutine directly

c

c sampled parameters are constant only.

c

---

subroutine addconstantpdf( name, conval)

---

c add constant PDF to sampled parameter database

c

c name = input, character\*60, name of pdf

c conval = input, double precision, constant value of pdf

c

---

subroutine addiconstantpdf( name, iconval)

---

c add integer constant PDF to sampled parameter database

c

c name = input, character\*60, name of pdf

c iconval = input, integer, constant value of pdf

c

---

subroutine adduniformpdf( name, xmin, xmax )

---

c add uniform PDF to database

c

c name = input, character\*60, name of pdf

c xmin = input, double precision

c xmax = input, double precision

c

---

subroutine addiuniformpdf( name, ixmin, ixmax )

---

c add integer uniform PDF to database

c

c name = input, character\*60, name of pdf

c ixmin = input, integer

c ixmax = input, integer

c

---

subroutine addloguniformpdf( name, xmin, xmax )

---

c add LogUniform PDF to database

```
c
c name = input, character*60, name of pdf
c xmin = input, double precision
c xmax = input, double precision
c
```

```
-----
      subroutine addnormalpdf( name,
&   valueat001quantile, valueat999quantile )
-----
```

```
c add normal PDF to database
```

```
c
c name = input, character*60, name of pdf
c valueat001quantile = input, double precision
c valueat999quantile = input, double precision
c
```

```
-----
      subroutine addlognormalpdf( name,
&   valueat001quantile, valueat999quantile )
-----
```

```
c add lognormal PDF to database
```

```
c
c name = input, character*60, name of pdf
c valueat001quantile = input, double precision
c valueat999quantile = input, double precision
c
```

```
-----
      subroutine addbetapdf( name,
&   alpha, beta, xmin, xmax )
-----
```

```
c add beta PDF to database
```

```
c
c name = input, character*60, name of pdf
c alpha = input, double precision
c beta = input, double precision
c xmin = input, double precision
c xmax = input, double precision
c
```

```
-----
      subroutine addlogbetapdf( name,
&   alpha, beta, xmin, xmax )
-----
```

```
c add logbeta PDF to database
```

```
c
c name = input, character*60, name of pdf
c alpha = input, double precision
c beta = input, double precision
c xmin = input, double precision
c xmax = input, double precision
c
```

```
-----
      subroutine addexponentialpdf( name, alam )
-----
```

```

=====
c add exponential PDF to database
c
c name = input, character*60, name of pdf
c alam = input, double precision
c
=====
      subroutine addfiniteexponentialpdf( name, alam, xmin, xmax )
=====
c add finite exponential PDF to database
c
c name = input, character*60, name of pdf
c alam = input, double precision, decay constant in exponential
c xmin = input, double precision, lower cut-off
c xmax = input, double precision, upper cut-off
c
=====
      subroutine addtriangularpdf( name,
      & xmin, xpeak, xmax )
=====
c add triangular PDF to database
c
c name = input, character*60, name of pdf
c xmin = input, double precision
c xpeak = input, double precision
c xmax = input, double precision
c
=====
      subroutine addlogtriangularpdf( name,
      & xmin, xpeak, xmax )
=====
c add logtriangular PDF to database
c
c name = input, character*60, name of pdf
c xmin = input, double precision
c xpeak = input, double precision
c xmax = input, double precision
c
=====
      subroutine newrealization()
=====
c Get new realization of inputs using random sampling
c
=====
      subroutine newlhssm( nbins )
=====
c new latin hypercube sampled parameter matrix
c that is stored in common blocks included in "sampler.i"
c
c nbins = input, integer, number of realizations
c
c          or LHS bins, or number of vectors

```

```

c          xlhs(nbin,nsp)
c
-----
c          integer function ispquery( namin )
-----
c          Get index for sampled parameter with name=namin
c
c          namin = input, character*60, name of sp as found in tpa.inp file
c          consequence modules must match name exactly
c          ispquery = output, integer, index of sampled parameter
c          which must be used to obtain the value of the
c          sample parameter. It is strongly suggested that
c          this integer index be stored in common block so that
c          this function need only be called once for each
c          parameter inside a consequence module
c
-----
c          integer function ispquerynostop( namin )
-----
c          Get index for sampled parameter with name=namin
c          if name not found, do NOT STOP, but return zero for index
c
c          namin = input, character*60, name of sp as found in tpa.inp file
c          consequence modules must match name exactly
c          ispquerynostop = output, integer,
c          =0 if not in database,
c          =index if name in database.
c          which must be used to obtain the value of the
c          sample parameter. It is strongly suggested that
c          this integer index be stored in common block so that
c          this function need only be called once for each
c          parameter inside a consequence module
c
-----
c          integer function isconstant( index )
-----
c          Query if parameter is a constant
c          Parameter specified by index.
c
c          index = input, index, index of sp found previously using ispquery
c
c          isconstant = output, integer,
c          =0 if not constant or iconstant or iflag
c          =1 if YES is a constant
c
-----
c          double precision function valuesp( ipdf )
-----
c          Get value of sampled parameter
c          where ipdf is index
c

```

```

c ipdf = input, integer, index of sampled parameter, this is
c   obtained using function ispquery
c valuesp = output, double precision, numeric value of parameter
c   value is changed ONLY when newrealization is called
c

```

---

```

integer function ivaluesp( ipdf)

```

---

```

c Get integer value of sampled parameter
c where ipdf is index
c

```

```

c ipdf = input, integer, index of sampled parameter, this is
c   obtained using function ispquery
c ivaluesp = output, integer, numeric value of parameter
c   value is changed ONLY when newrealization is called
c

```

---

```

subroutine addhazardcurve( name, nbin, amag, period )

```

---

```

c add HazardCurve to database
c

```

```

c name = input, character*60, name of hazard curve
c nbin = input, integer, number of hazard bins
c amag(nbin) = input, double precision, magnitude of events
c period(nbin) = input, double precision , return period
c   in units of [yr] for events
c

```

---

```

subroutine samplehazardcurve( name, maxevents, timemax,
& numerofevents, timeofevents, typeofevents )

```

---

```

c draw realization of future events from HazardCurve
c

```

```

c name = input, character*60, name of hazard curve
c maxevents = input, integer, maximum number of events,
c   used to dimension arrays
c timemax = input, double precision, time period of interest (TPI)
c numerofevents = output, integer, number of events
c   NOTE: numerofevents <= maxevents
c   if numerofevents = maxevents, then will print
c   WARNING message that maxevents should be increased
c timeofevents(maxevents) = output, double precision, time [yr] of
c   series of events
c typeofevents(maxevents) = output, double precision, type of
c   (or magnitude of) events
c

```

---

```

subroutine lhsnew( maxnbin, maxnsp, nbin, nsp, x )

```

---

```

c create new LHS quantile sampled matrix,

```

```

c this subroutine is generic, and not an integral part of TPA 3.0
c Ex: it does not need or use "sampler.i" include file
c
c maxnbin = input, integer, maximum number of vectors or bins
c           used in LHS, to dimension arrays
c maxnsp = input, integer, maximum number of sampled parameters
c           used to dimension arrays
c nbin   = input, integer, actual number of LHS bins
c nsp    = input, integer, actual number of sampled parameters
c x(maxnbin,maxnsp) = input/output, double precision, array of
c                   quantiles sampled from U[0,1] that are randomized for LHS
c

```

---

```

subroutine printtitlesp( iunitcp, iunitsp )

```

---

```

c print names of constant & sampled parameters
c names are character*60 titles of parameters
c
c iunitcp = input, integer, unit number having open file to dump titles
c           for constant parameters
c
c iunitsp = input, integer, unit number having open file to dump titles
c           for sampled parameters
c

```

---

```

subroutine printvaluessp( iunit, ir )

```

---

```

c print values of all sampled parameters into file opened
c with unit number equal iunit
c
c this will print constants, iconstants, and iflag parameters
c as if they are sampled parameters
c
c iunit = input, integer, unit number having open file to dump titles
c ir    = input, integer, realization number
c

```

---

```

subroutine writesnllhsinp( iunit )

```

---

```

c write input file for LHS program written by SNL
c
c iunit = input, integer, unit number having open file to write to
c

```

---

```

subroutine checkspname( name )

```

---

```

c check name to see if already defined in sampled parameter database
c
c name = input, character*60, name of pdf
c

```

---

```
subroutine adduserdist( name, n, v )
```

```
-----  
c add user-supplied data PDF to sampled parameter database  
c  
c name = input, character*60, name of pdf  
c n   = input, integer, number of user supplied data  
c v(n) = input, double precision, values of user supplied data,  
c      all data is equal probability (p=1/n)  
c
```

```
-----  
subroutine addcorrel( name1i, name2i, c )  
-----
```

```
c add rank correlation for 2 sampled parameters  
c  
c name1i = input, character*60, name of first parameter  
c name2i = input, character*60, name of second parameter  
c c      = input, double precision, rank correlation  
c
```

•

## C.5 RAN utility module (file=ran.f)

c Group of random number subroutines used by sampled parameter  
c utility module, by Randall D. Manteufel, January 27, 1997

---

subroutine setran( aseed1 )

---

c set seed for random number generator

c

c aseed1 = input, double precision, seed value

c recommend value for aseed be between 1.0d8 and 1.0d9 when  
c working in double precision.

c if changed to single precision, recommend 1.e3 to 1.e5

c

---

function ran1( )

---

c random number generator.

c based on congruential generator described in "Stochastic Simulation"

c by Brian D. Ripley, 1987, John Wiley & Sons

c see page 20, equation (1) and Table 2-4 on pg 39. Used 4th algorithm  
c in table.

c

c ran1 = output, double precision, random number between 0.0 and 1.0

c

---

function ranb(alpha,beta,xmin,xmax)

---

c random sample from beta pdf

c Algorithm from: Probability Concepts in Engineering Planning and Design

c Volume II: Decision, Risk, and Reliability

c Alfredo H-S. Ang and Wilson H. Tang

c John Wiley & Sons, New York, 1984, pp. 285-287

c

c alpha = input, double precision, parameter in beta distribution

c beta = input, double precision, parameter in beta distribution

c xmin = input, double precision, minimum value of sampled parameter

c xmax = input, double precision, maximum value of sampled parameter

c ranb = output, double precision, value of sampled parameter

c

---

function ranlb(alpha,beta,xmin,xmax)

---

c random sample from logbeta pdf

c

c see description of ranb input/output parameters

c

---

function rant(xmin, xpeak, xmax)

---

c random sample from triangular pdf

c  
c xmin = input, double precision, minimum range of sampled parameter  
c xpeak = input, double precision, value of sampled parameter where  
c       triangular pdf has peak  
c xmax = input, double precision, maximum value of range of sampled parameter  
c rant = output, double precision, value of sample parameter  
c

---

function ranlt(xmin, xpeak, xmax)

---

c random sample from logtriangular pdf  
c  
c see description of rant input/output parameters  
c

---

function rane(alam)

---

c random sample from exponential  
c pdf =  $\text{alam} * \text{EXP}(-\text{alam} * t)$   
c  
c alam = input, double precision, input parameter for exponential pdf  
c rane = output, double precision, sampled parameter from exponential pdf  
c       Note:  $0.0 < \text{rane} < \text{Infinity}$   
c

---

function ranfe(alam, xmin, xmax)

---

c random sample from finite exponential  
c pdf =  $\text{alam} * \text{EXP}(-\text{alam} * t)$   
c  
c alam = input, double precision, input parameter for exponential pdf  
c xmin = input, double precision, lower limit for exponential pdf  
c xmax = input, double precision, upper limit for exponential pdf  
c rane = output, double precision, sampled parameter from exponential pdf  
c       Note:  $\text{xmin} < \text{rane} < \text{xmax}$   
c

---

function ranu(xlow, xhigh)

---

c random sample from uniform pdf  
c  
c xlow = input, double precision, minimum range of pdf  
c xhigh = input, double precision, maximum range of pdf  
c ranu = output, double precision, random sampled parameter  
c

---

function ranlu(xlow, xhigh)

---

c random sample from loguniform pdf  
c  
c see description of ranu input/output parameters

```

c
=====
function iranu(ilow, ihigh)
=====
c integer random sample from uniform pdf
c
c ilow = input, integer, low value of pdf
c ihigh = input, integer, high value of pdf
c iranu = output, integer, sampled value
c     Note:  ilow <= iranu <= ihigh
c
=====
function rann( v001q, v999q )
=====
c random sample from normal pdf
c
c v001q = input, double precision, x value at 0.001 quantile
c     probability of cdf for normal distribution.
c     This is -3.090232306167814 standard deviations below mean.
c v999q = input, double precision, x value at 0.999 quantile
c     probability of cdf for normal distribution.
c     This is +3.090232306167814 standard deviations above mean.
c rann = output, double precision, sampled value from normal pdf
c
=====
function ranln( v001q, v999q )
=====
c random sample from lognormal pdf
c
c see description of rann input/output parameters
c
=====
function gasdev()
=====
c random sample from unit normal pdf
c having zero mean and unit variance
c from Numerical Recipes, W.H. Press, B.P. Flannery, S.A. Teukolsky
c and W.T. Vetterling. 1990. pg 203
c
c gasdev = output, double precision, sampled value from pdf
c
=====
function quantu( al, ah, p )
=====
c Obtain parameter by inverting CDF at given probability level
c
c al   = input, double precision, low value for uniform distribution
c ah   = input, double precision, high value for uniform distribution
c p    = input, double precision, quantile level
c quantu = output, double precision, value between al (low) and ah (high)
c     sampled at quantile level p

```

```

c
c=====
c      function quantu( ial, iah, p )
c=====
c Obtain parameter by inverting CDF at given probability level
c
c ial   = input, integer, low value for uniform distribution
c iah   = input, integer, high value for uniform distribution
c p     = input, double precision, quantile level
c quantu = output, integer, value between ial (low) and iah (high)
c       sampled at quantile level p
c
c=====
c      function quantl( n, x, iflag, ix, p )
c=====
c invert cdf at "p" probability ( $0 < p < 1$ )
c based on list of equal probability x-values,
c will interpolate between values of set, but will not extrapolate
c either above or below set.
c   if  $p < 1/(2*n)$ , then  $quantl=x(1)$ 
c   if  $p > 1 - 1/(2*n)$ , then  $quantl=x(n)$ 
c
c for example, if  $n=5$ ,  $x=\{1,2,3,4,5\}$ ,
c            $p=.50$ , then  $quantl=3.0$ 
c            $p=.30$ , then  $quantl=2.0$ 
c            $p=.25$ , then  $quantl=1.75$ 
c            $p = 0.0001$ , then  $quantl=1$ 
c for example, if  $n=4$ ,  $x=\{1,2,3,4\}$ ,
c            $p=0.50$ , then  $quantl=2.5$ 
c            $p=.333$ , then  $quantl=1.832$ 
c            $p=.125$ , then  $quantl=1.0$ 
c            $p = 0.0001$ , then  $quantl=1$ 
c for example, if  $n=4$ ,  $x=\{1, 100, 200, 1000\}$ ,
c            $p=0.50$ , then  $quantl=150$ 
c            $p = 0.0001$ , then  $quantl=1$ 
c
c n     = input, integer, length of array values, must be  $\geq 1$ 
c x[n]  = input, double precision, array of values
c       values can be repeated
c iflag = input, integer, =0 if not sorted
c       =1 if sorted
c       if iflag = 1, then quantl believes the caller and
c       will not check to be sure it is sorted
c       if in doubt, set iflag =0
c ix[n] = integer, input, array used only if not SORTED
c p     = input, double precision, quantile level,  $0 < p < 1$ 
c quantl = output, double precision, value from list (array) at quantile level
c
c=====
c      function ranset( n, x )
c=====

```

c randomly select one value of a set of values  
 c all values have equal probability of being selected  
 c will return only values in the set  
 c set need not be sorted  
 c  
 c n = input, integer, length of array values  
 c x(n) = input, double precision, array of values  
 c ranset = output, double precision, random value from list (array)

---

function quantset( n, x, p )

---

c select one value of a set of values  
 c all values have equal probability of being selected  
 c will return only values in the set  
 c set need not be sorted  
 c  
 c n = input, integer, length of array values  
 c x(n) = input, double precision, array of values  
 c p = input, double precision, quantile level,  $0 < p < 1$   
 c quantset = output, double precision, value from list (array)  
 c at quantile level p

---

function quantn( v001q, v999q, p )

---

c quantile of Normal distribution  
 c  
 c v001q = input, double precision, x value at 0.001 quantile  
 c probability of cdf for normal distribution.  
 c This is -3.090232306167814 standard deviations below mean.  
 c v999q = input, double precision, x value at 0.999 quantile  
 c probability of cdf for normal distribution.  
 c This is +3.090232306167814 standard deviations above mean.  
 c p = input, double precision, value of cdf, must be between 0 and 1  
 c quantn = output, double precision, value of pdf at quantile=p

---

function cdfn( amu, sig, x )

---

c Cumulative Distribution Function (cdf) for Normal distribution  
 c  
 c amu = input, double precision, mean of normal distribution  
 c sig = input, double precision, standard deviation  
 c x = input, double precision, value of parameter  
 c cdfn = output, double precision, value of cdf at parameter=x

---

function quantb( alpha, beta, xmin, xmax, p )

---

c quantile of beta distribution

c  
 c alpha = input, double precision, parameter in beta distribution  
 c beta = input, double precision, parameter in beta distribution  
 c xmin = input, double precision, minimum value of sampled parameter  
 c xmax = input, double precision, maximum value of sampled parameter  
 c p = input, double precision, quantile level at which to invert CDF  
 c quantb= output, double precision, value of sampled parameter at specified  
 c quantile level  
 c

---

function betacdf(a,b,x)

---

c evaluate CDF of standard beta function

c  
 c a = input, double precision, parameter in Beta Function  
 c b = input, double precision, parameter in Beta Function  
 c x = input, double precision, independent variable  
 c betacdf = output, double precision, CDF of beta fcn  
 c

---

function betaspecial(x)

---



---

function betapdf(a,b,x)

---

c evaluate PDF of standard beta function

c  
 c a = input, double precision, parameter in Beta Function  
 c b = input, double precision, parameter in Beta Function  
 c x = input, double precision, independent variable  
 c betapdf = output, double precision, PDF of beta fcn  
 c

---

function quantlb( alpha, beta, xmin, xmax, p )

---

c quantile of log beta distribution

c  
 c alpha = input, double precision, parameter in beta distribution  
 c beta = input, double precision, parameter in beta distribution  
 c xmin = input, double precision, minimum value of sampled parameter  
 c xmax = input, double precision, maximum value of sampled parameter  
 c ranb = output, double precision, value of sampled parameter  
 c

---

function quantt(xmin, xpeak, xmax, p)

---

c quantile from triangular pdf

c  
 c xmin = input, double precision, minimum range of sampled parameter  
 c xpeak = input, double precision, value of sampled parameter where  
 c triangular pdf has peak  
 c

c xmax = input, double precision, maximum value of range of sampled parameter  
c p = input, double precision, quantile level  
c quantt = output, double precision, value of sample parameter  
c at quantile level of CDF  
c

---

function quantlt(xmin, xpeak, xmax, p)

---

c quantile from log triangular pdf

c  
c xmin = input, double precision, minimum range of sampled parameter  
c xpeak = input, double precision, value of sampled parameter where  
c triangular pdf has peak  
c xmax = input, double precision, maximum value of range of sampled parameter  
c p = input, double precision, quantile level  
c quantlt = output, double precision, value of sample parameter  
c at quantile level of CDF  
c

---

function quantln( v001qin, v999qin, p)

---

c quantile from log normal pdf

c  
c v001qin = input, double precision, x value at 0.001 quantile  
c probability of cdf for normal distribution.  
c This is -3.090232306167814 standard deviations below mean.  
c v999qin = input, double precision, x value at 0.999 quantile  
c probability of cdf for normal distribution.  
c This is +3.090232306167814 standard deviations above mean.  
c p = input, double precision, value of cdf, must be between 0 and 1  
c quantln = output, double precision, value of pdf at quantile=p  
c

---

function quante(alam,p)

---

c quantile from exponential

c pdf =  $\text{alam} * \text{EXP}(-\text{alam} * t)$ , range of t = {0, Infinity}  
c cdf =  $1.0 - \text{EXP}(-\text{alam} * t)$   
c

c alam = input, double precision, input parameter for exponential pdf  
c p = input, double precision, quantile level  
c quante = output, double precision, quantile from exponential pdf  
c

---

function quantfe(alam, xminin, xmaxin, p)

---

c quantile from finite exponential

c pdf =  $\text{alam} * \text{EXP}(-\text{alam} * t)$   
c

c alam = input, double precision, input parameter for exponential pdf  
c xminin = input, double precision, lower limit for exponential pdf

c xmaxin = input, double precision, upper limit for exponential pdf  
c p = input, double precision, quantile level  
c quantfe = output, double precision, quantile from finite exponential pdf  
c

---

c  
function quantlu(xmin, xmax, p)

---

c  
c quantile from log uniform pdf

c  
c xmin = input, double precision, minimum range of sampled parameter  
c xmax = input, double precision, maximum value of range of sampled parameter  
c p = input, double precision, quantile level  
c quantlu = output, double precision, value of sample parameter  
c at quantile level of CDF  
c

## C.6 MODULE VARIABLE utility module (file=mv.f)

c Module Variable utility module  
c by Randall D. Manteufel, January 27, 1997

---

subroutine newmvdb( )

---

c called only by TPA code

c  
c initialize new database for module variables

c  
c module variables can be only "constant"

---

integer function iaddconsmv( name )

---

c add constant variable to database

c NOTE: must call imvquery to get index number for this mv  
c NOTE: must call setconsmv to set value of variable

c  
c name = input, character\*60, name of module variable  
c iaddconsmv (=iowner) = output, integer, special index for module  
c variable so that only the "owner" who knows this index  
c can change the value of the variable.  
c Others can query.

---

integer function imvquery( name )

---

c Get index for module variable with "name"

c  
c name = input, character\*60, name of module variable as found in tpa.inp file  
c consequence modules must match name exactly  
c imvquery = output, integer, index of module variable  
c which must be used to obtain the value(s) of the  
c module variable. It is strongly suggested that  
c this integer index be stored in common block so that  
c this function need only be called once for each  
c variable inside a consequence module

---

subroutine setconsmv( imv, iowner, consv )

---

c set constant module variable to value=consv

c  
c imv = input, integer, index to module variable  
c iowner = input, integer, module index needed to set/change value of  
c module variable.  
c Only the owner of variable can control the value of  
c this variable.  
c consv = input, double precision, constant value

c

---

function valueconsmv( imv )

---

c

c Get value of module variable  
c where imv is index for module variable

c

c imv = input, integer, index of module variable, this is obtained using  
c function "imvquery"  
c valueconsmv = output, double precision, numeric value of module variable  
c module variable is type = "constant"

c

---

subroutine printtitlesmv( iunit )

---

c

c print number of module variables, and their character\*60 titles  
c this will be printed into file opened with unit number equal iunit

c

c iunit = input, integer, unit number having open file to dump titles

c

c

---

subroutine printvaluesmv( iunit, ir )

---

c

c print values of all module variables into file opened  
c with unit number equal iunit

c

c iunit = input, integer, unit number having open file to dump values

c ir = input, integer, realization number

c

## C.7 INVENT utility module (file=invent.f)

```
c invent utility module
c by Randall D. Manteufel, January 27, 1997
c
c-----
c      subroutine setage( timeatemplacement )
c-----
c set age of HLW at time of emplacement.
c age is measured in [yr] for reactor to repository
c The age affects inventories of isotopes and
c thermal output of waste.
c
c timeatemplacement = input, double precision, time in years from
c reactor at which fuel is emplaced in repository
c all subsequent times are considered from emplacement
c Setage does not have to be called.
c If not called, then default age is 26 yr.
c The numerics are such that it can be called with age > 1 yr.
c
c-----
c      subroutine decay43mol( time, amolepermtu )
c-----
c This subroutine returns all radioisotopes inventories [mole/MTU]
c at "time" [yr] after emplacement,
c assuming only decay [no release] of isotopes from source.
c
c time = input, double precision, time [yr] measured after emplacement
c amolepermtu(43) = output, double precision, inventory [mole/MTU]
c for each isotope at time
c
c (d N[i])/(dt) = -dldecay[i] * N[i]
c (d N[i+1])/(dt) = -dldecay[i+1] * N[i+1] +dldecay[i] * N[i]
c (d N[i+2])/(dt) = -dldecay[i+2] * N[i+2] +dldecay[i+1] * N[i+1]
c (d N[i+3])/(dt) = -dldecay[i+3] * N[i+3] +dldecay[i+2] * N[i+2]
c
c dldecay[i] = ln(2) / halflife(i)
c "d" = double
c "l" = lambda, transformation constant
c "decay" = due to radioactive decay
c
c Order of Nuclides:
c
c 1- U238
c 2- Cm246
c 3- Pu242
c 4- Am242m
c 5- Pu238
c 6- U234
c 7- Th230
c 8- Ra226
```

c 9- Pb210  
 c 10- Cm243  
 c 11- Am243  
 c 12- Pu239  
 c 13- U235  
 c 14- Pa231  
 c 15- Ac227  
 c 16- Cm245  
 c 17- Pu241  
 c 18- Am241  
 c 19- Np237  
 c 20- U233  
 c 21- Th229  
 c 22- Cm244  
 c 23- Pu240  
 c 24- U236  
 c 25- U232  
 c 26- Sm151  
 c 27- Cs137  
 c 28- Cs135  
 c 29- I129  
 c 30- Sn126  
 c 31- Sn121m  
 c 32- Ag108m  
 c 33- Pd107  
 c 34- Tc99  
 c 35- Mo93  
 c 36- Nb94  
 c 37- Zr93  
 c 38- Sr90  
 c 39- Se79  
 c 40- Ni63  
 c 41- Ni59  
 c 42- Cl36  
 c 43- C14

c  
c The chains that need to be tracked are:

c  
c Chain #1

c  
c 2---3---1---6---7---8---9

c |  
c 4--5--/

c  
c Chain #2

c  
c 10--12--13--14--15

c |  
c 11--/

c

```

c
c Chain #3
c
c 16--17--18--19--20--21
c
c
c Chain #4
c
c 22--23--24
c
c Note: U-232 does not appear in any chain and is treated as
c Fission Product
c

```

---

```

c subroutine decayremove43mol( dt, dlremove, dnt1, dnt2 )

```

---

```

c Solves for inventories of 43 isotopes tracked in HLW
c accounting for both decay of isotopes and removal of isotopes
c [For example, release is a physical removal from waste package]
c
c dt = input, integer, time step, = endtime - starttime [yr]
c dlremove(43) = input, double precision, array of physical
c removal constant,
c dlremove has units of [1/yr]
c dnt1(43) = input, double precision, array of initial inventories
c in units of [mole]
c dnt2(43) = output, double precision, array of final inventories
c in units of [mole]
c
c (d N[i])/(dt) = -dldecay[i] * N[i]
c -dlremove[i] * N[i]
c (d N[i+1])/(dt) = -dldecay[i+1] * N[i+1] +dldecay[i] * N[i]
c -dlremove[i+1] * N[i+1]
c (d N[i+2])/(dt) = -dldecay[i+2] * N[i+2] +dldecay[i+1] * N[i+1]
c -dlremove[i+2] * N[i+2]
c (d N[i+3])/(dt) = -dldecay[i+3] * N[i+3] +dldecay[i+2] * N[i+2]
c -dlremove[i+3] * N[i+3]
c

```

---

```

c subroutine chains(maxnt, nt, time, dlt, dlr, dnt1, dn )

```

---

```

c Solves for inventories of 43 isotopes tracked in HLW
c accounting for both decay of isotopes and removal of isotopes
c
c the decay accounts for chains
c maxnt = input, integer, maximum number of time to dimension arrays
c nt = input, integer, number of times
c time(nt) = input, double precision, array of times
c All times in units of [yr]
c to match units of dlt & dlr
c dlt(43) = input, double precision, array of total

```

```

c          total removal constant which equals SUM of
c          (1) radioactive decay and
c          (2) all sources of physical removal such as
c          leaching into ground water
c          dlt has units of [1/yr]
c dlr(43) = input, double precision, array of removal constants
c          for radioactive decay,
c          this is  $\lambda = \ln(2) / \text{half-life}$ 
c          half-life in units of [yr]
c dnt1(43) = input, double precision, initial amounts for each isotope
c          in the chain, this is measured in units of [mole]
c dn(maxnt,43) = output, double precision, final amounts of isotopes
c          in units of [mole] in chain at each time step
c          Note:  $\text{dn}(1,\text{inucs}) = \text{dnt1}(\text{inucs})$ 
c

```

---

```

c          subroutine chainsolver(len, maxnt, nt, tim, dnt1, dlt, dlr, dn)

```

---

```

c solves for one chain of isotopes, accounting for radioactive decay
c and physical removal from system.
c

```

```

c solves for time-dependent isotope inventory
c when there is both physical removal and radioactive decay
c in a chain:
c

```

```

c  $(d N[i]) / (dt) = -dlt * N[i]$ 
c  $(d N[i+1]) / (dt) = -dlt * N[i+1] + dlr * N[i]$ 
c  $(d N[i+2]) / (dt) = -dlt * N[i+2] + dlr * N[i+1]$ 
c etc.
c

```

```

c len = input, integer, the number of radioisotopes in the chain
c maxnt = input, integer, the maximum number of times that was
c          used to dimension arrays
c nt = input, integer, the number of times to calculate inventory at
c tim(nt) = input, double precision, array of time values
c dnt1(len) = input, double precision, the initial amounts
c          for each isotope in the chain [mole]
c          corresponds to time=tim(1)
c dlt(len) = input, double precision, the total removal rate
c          for each isotope in the chain, has units [1/yr]
c          Total = Physical + Radioactive
c dlr(len) = input, double precision, the radioactive decay
c          rate for each isotope in the chain [1/yr]
c          the 1st decays into the 2nd,
c          the 2nd decays into the 3rd, etc
c dn(maxnt,len) = output, double precision, the inventories
c          [mole] of each isotope in the chain at each time
c          Note:  $\text{dn}(1,\text{len}) = \text{dnt1}(\text{len})$ 
c

```

---

```

c          double precision function dmyexp(x)

```

---

```

=====
c MY exponential function, so that will set
c   exp( large negative number ) = 0.0
c   and give Warning for exp( large positive number )
c
c This function is used to avoid underflow warnings from SUN Fortran
c compiler.
c
c x = input, double precision, input value
c dmyexp = output, double precision, output value
c
=====
integer function indexperiso( name )
=====
c returns index based on matching isotope "name"
c
c name = input, character*6, name of isotope
c indexperiso = output, integer, index used to identify isotope
c   of interest
c
=====
integer function indexperisonostop( name )
=====
c returns index based on matching isotope "name"
c
c name = input, character*6, name of isotope
c indexperiso = output, integer, index used to identify isotope
c   of interest
c   Will not stop if "name" not found, but will return
c   indexperisonostop = 0
c
=====
character*2 function nameelem( index )
=====
c returns element name given isotope index,
c for example, will return 'Am'
c given index = either 4, 11, or 18
c corresponding to isotopes Am242m, Am243 or Am241
c
c index = input, integer, index to isotope of interest
c nameelem = output, character*2, name of element
c
=====
character*6 function nameiso( index )
=====
c returns name of isotope
c
c index = input, integer, index to isotope of interest
c nameiso = output, character*6, name of isotope
c
=====

```

function epalimperiso( index )

=====

c get EPA normalizing limits for isotopes,  
c this is for old 'release' limit  
c  
c index = input, integer, index to isotope of interest  
c epalimperiso = output, double precision, EPA release limit  
c for isotope, in [Ci/MTU]  
c

function activityperiso( index )

=====

c returns activity of isotope [Ci/mole]  
c  
c index = input, integer, index used to identify isotope  
c of interest  
c activityperiso = output, double precision,  
c activity of isotope [Ci/mole]  
c

function wmoleperiso( index )

=====

c molecular weight of radioisotope [g/mol]  
c  
c index = input, integer, index used to quickly identify isotope  
c of interest  
c wmoleperiso = output, double precision, molecular weight  
c of isotope [g/mol]  
c

function halflifeperiso( index )

=====

c return halflife of radioisotope [yr]  
c  
c index = input, integer, index used to quickly identify isotope  
c of interest  
c halflifeperiso = output, double precision, halflife [yr]  
c

c Following is a comparison of the half-lives used in INVENT  
c compared to the half-lives found in two references -  
c 1992 Chart of the Nuclides - Strasbourg and the  
c 1991-2 CRC Handbook of Chemistry and Physics  
c Invent halflife values from DOE/SNL TSPA-93  
c

c The comparison shows good agreement in halflife values  
c and builds confidence in the data.

c

Source	
-----	
c	1992 chart of
c	1991-2 CRC Handbook

c Number	Nuclide	nuclides	INVENT	of Chemistry + Physics
c 1	U238	4.468e9	4.468e9	4.46e9
c 2	Cm246	4753	4731	4780
c 3	Pu242	3.73e5	3.869e5	3.76e5
c 4	Am242m	142	152	141
c 5	Pu238	87.75	87.74	87.74
c 6	U234	2.455e5	2.445e5	2.45e5
c 7	Th230	7.54e4	7.70e4	7.54e4
c 8	Ra226	1600	1600	1599
c 9	Pb210	22.3	22.3	22.6
c 10	Cm243	28.5	28.5	28.5
c 11	Am243	7362	7380	7370
c 12	Pu239	2.42e4	2.406e4	2.411e4
c 13	U235	7.04e8	7.038e8	7.04e8
c 14	Pa231	3.282e4	3.277e4	3.25e4
c 15	Ac227	21.773	21.77	21.77
c 16	Cm245	8511	8499	8530
c 17	Pu241	14.353	14.40	14.4
c 18	Am241	432.1	432.2	432.2
c 19	Np237	2.14e6	2.14e6	2.14e6
c 20	U233	1.59e5	1.585e5	1.59e5
c 21	Th229	7686	7339	7900
c 22	Cm244	18.077	18.11	18.11
c 23	Pu240	6555	6537	6537
c 24	U236	2.342e7	2.341e7	2.34e7
c 25	U232	69.9	72	68.9
c 26	Sm151	90	89.99	90
c 27	Cs137	30.254	30.0	30.3
c 28	Cs135	2.3e6	2.3e6	2.3e6
c 29	I129	1.6e7	1.57e7	1.7e7
c 30	Sn126	1e5	1e5	1e5
c 31	Sn121m	55	49.97	55
c 32	Ag108m	127	127	130
c 33	Pd107	6.5e6	6.496e6	6.5e6
c 34	Tc99	2.1e5	2.13e5	2.13e5
c 35	Mo93	3500	3498	3500
c 36	Nb94	2e4	2.03e4	2.4e4
c 37	Zr93	1.53e6	1.53e6	1.5e6
c 38	Sr90	28.78	29.12	29.1
c 39	Se79	6.5e4	6.496e4	6.5e4
c 40	Ni63	99.6	92	100
c 41	Ni59	7.6e4	8.0e4	7.6e4
c 42	Cl36	3.01e5	3.01e5	3.01e5
c 43	C14	5730	5729	5715

c

---

function alambdaperiso( index )

---

c return "lambda" radioactive decay transformation constant [1/yr]

c

c index = input, integer, index used to quickly identify isotope

c of interest  
c alambdaperiso = output, double precision, transformation constant  
c [1/yr]  
c

---

function cipermtuat10periso( index )

---

c return reference isotope inventory in units of [Ci/MTU] for  
c representative waste at 10 yr from reactor  
c "representative waste" is blend of spent fuel and other HLW  
c and is discussed in report:  
c  
c INVENT: A Module for the Calculation of Radioisotope Inventories  
c Software Description, and User Guide. by A.S. Lozano, H. Karimi,  
c J.P. Cornelius, R.D. Manteufel, and R.W. Janetzke. 1994.  
c CNWRA 94-016.  
c  
c index = input, integer, index used to quickly identify isotope  
c of interest  
c cipermtuat10periso = output, double precision, reference  
c inventory at 10 yr from reactor [Ci/MTU]  
c

---

function amolepermtuatemplacperiso( index )

---

c inventory of waste [mole/MTU] at time of emplacement  
c time of emplacement is controlled by "setage"  
c if setage not called, then default time from reactor to emplacement  
c is 26 yr  
c  
c index = input, integer, index used to quickly identify isotope  
c of interest  
c amolepermtuatemplacperiso = output, double precision, reference  
c inventory at time of emplacement [mole/MTU]  
c default is at 26 yr from reactor if setage not called  
c

---

function cipermtuatemplacperiso( index )

---

c inventory of waste [Ci/MTU] at time of emplacement  
c time of emplacement is controlled by "setage"  
c if setage not called, then default time from reactor to emplacement  
c is 26 yr  
c  
c index = input, integer, index used to quickly identify isotope  
c of interest  
c cipermtuatemplacperiso = output, double precision, reference  
c inventory at time of emplacement [Ci/MTU]  
c default is at 26 yr from reactor if setage not called  
c

---

subroutine newinventdb()

---

c  
c The USER should never call this subroutine.  
c It will be called automatically when invent data is needed  
c  
c initialize new inventory database  
c default age is 26 yr from reactor  
c

---

function qpermtu( time )

---

c  
c thermal output of blended HLW, in units of [W/MTU]  
c  
c time = input, double precision, time after emplacement  
c           measured in units of [yr]  
c  
c qpermtu = output, double precision, heat output of waste  
c           measured in units of [W/MTU]  
c  
c based on blend of 65% PWR with 42 GWd/MTU burnup  
c           35% BWR with 32 GWd/MTU burnup  
c  
c therefore average burnup is 38.5 GWd/MTU  
c  
c

## C.8 SUBAREA utility module (file=subarea.f)

```
c subarea utility module
c by Randall D. Manteufel, January 27, 1997
c-----
c      subroutine ssadb( nsa, xy, amtupersa, nwppersa )
c-----
c set (or initialize) entire subarea in database
c this is only called ONCE.
c
c nsa      = input, integer, number of subareas in repository
c xy(2,4,nsa) = input, double precision, locations of vertices of
c            quadrilateral subareas.
c            All measurements in Universal Transverse Mercator (UTM)
c            in units of [m].
c            x-direction: UTM Easting [m]
c            y-direction: UTM Northing [m]
c amtupersa[nsa] = input, double precision, amount of MTU waste in
c            each subarea
c nwppersa[nsa] = input, integer, number of WP in each subarea
c
c-----
c      subroutine gnsa( nsa )
c-----
c get total number of subareas in database
c
c nsa    = input, integer, total number for the subareas
c
c-----
c      subroutine gsarea( isa, area )
c-----
c get magnitude of area [m^2] of subarea
c
c isa    = input, integer, number for the subarea
c area   = output, double precision, area of subarea
c            in units of [m^2]
c
c-----
c      subroutine gsaxy( isa, xy )
c-----
c get vertice locations for subarea number=isa
c
c isa    = input, integer, number for the subarea
c xy[2,4] = output, double precision, locations of
c            vertices of quadrilateral
c
c-----
c      subroutine gsaxym( isa, xymiddle )
c-----
c get location of middle for subarea number=isa
c
```

```

c isa      = input, integer, number for the subarea
c xymiddle[2] = output, double precision, location of
c           center of quadrilateral
c
-----
      subroutine gsamtu( isa, amtupersa )
-----
c get amount of waste [MTU] emplaced in subarea number=isa
c
c isa      = input, integer, number for the subarea
c amtupersa = output, double precision, amount of MTU waste in subarea
c
-----
      subroutine gsanwp( isa, nwppersa )
-----
c get number of WP emplaced in subarea number=isa
c
c isa      = input, integer, number for the subarea
c nwppersa = output, integer, number of WP in subarea
c
-----
      subroutine qphitsa( xyp, isa, iflag )
-----
c query if point hits subarea
c
c xyp[2] = input, double precision, location of point
c isa    = input, integer, number of subarea to be checked
c iflag  = output, integer, =1 if point in or on edge of subarea,
c         =0 if not
c
-----
      subroutine qchitsa( xyp, radius, isa, iflag, areainsa )
-----
c query if circle hits subarea
c Because circles are small in comparison with subarea, the center of
c the circle must be inside subarea.
c
c xyp[2] = input, double precision, location of center of circle
c radius = input, double precision, radius of circle
c isa    = input, integer, number of subarea to be checked
c iflag  = output, integer, =1 if hist subarea, =0 if not
c areainsa = output, double precision, area of subarea hit by circle,
c         if iflag=1, areainsa = Pi radius^2
c         if iflag=0, areainsa = 0.0d0
c
-----
      subroutine qlhitsa( xyp1, xyp2, isa, iflag, alengthinsa )
-----
c query if line hits subarea
c
c xyp1[2] = input, double precision, first point on line

```

```

c xyp2[2] = input, double precision, second point on line
c isa = input, integer, number of subarea to be checked
c iflag = output, integer, =1 if line hits subarea, =0 if not
c alengthinsa = output, double precision, length of line in subarea
c =0.0d0 if iflag = 0
c

```

```

-----
c      subroutine linehitline( xyp11, xyp211,
&          xyp112, xyp212, c1, c2 )

```

```

c determine if lines cross
c
c xyp11(2) = coordinates of 1st end point of 1st line
c xyp211(2) = coordinates of 2st end point of 1st line
c xyp112(2) = coordinates of 1st end point of 2st line
c xyp212(2) = coordinates of 2st end point of 2st line
c c1 = parametrization value for where 1st line is hit by 2nd
c c2 = parametrization value for where 2nd line is hit by 1st
c

```

```

-----
c      subroutine solve2x2( a, b, x )

```

```

c solve 2x2 system of equations using Cramer's rule: [A] {x} = {b}
c
c a(2,2) = input, double precision, matrix
c b(2) = input, double precision, right hand side vector
c x(2) = output, double precision, solution vector
c

```

```

-----
c      subroutine checkin( xyp, xy, iflag)

```

```

c This subroutine determines whether a point (xyp(1),xyp(2)) is
c inside a given region defined by [x(1),y(1)], [x(2),y(2)],
c [x(3),y(3)], and [x(4),y(4)].
c
c xyp[2] = input, double precision, location of point
c xy[2,4] = input, double precision, locations of corners of quadrilateral
c iflag = output, integer, =0 if point outside region
c =1 if point inside region
c if point is on edge of box, it will say is inside region
c

```

```

-----
c      double precision function fcnxy( n, i, c, xy )

```

```

c function used to determine {c1, c2} coordinates of point {x,y}
c in coordinate system established by quadrilateral element
c
c n = input, integer, =2
c i = input, integer = {1,2} indicating either x or y
c c(2) = input, double precision, local coordinates in

```

```

c      system for quad element
c xy(2,5) = input, double precision, global coordinates of
c      1) four points of quadrilateral element, AND
c      2) point for which c-coordinates are being sought
c      hence there are 5 {x,y} data
c
-----
c      subroutine root( n, x, eps, ifail, x2, f, df, fcn, dat,
c      & mknt, iknt )
-----
c find root of system of equations using Newton-Raphson Iteration
c
c {f({x})} = {0}, where {0}, {x}, and {f} are vectors
c
c n = input, integer, number of unknown x's
c x(n) = input/output, double precision, initial guess at root of system
c      and on OUTPUT is the solution
c eps = input, double precision, error tolerance, will stop
c      iterating if absolute value of each component of f <= eps
c      AND if most recent change in each component of x <= eps
c ifail = output, integer, flag indicating failure to converge
c      =0 if converged
c      =1 if failed
c x2(n) = input, double precision, work space
c f(n) = input, double precision, work space
c df(n,n) = input, double precision, work space
c fcn = input, function, function subroutine that can be called
c      to evaluate f(x) for specific equation
c      used as:
c      fcn(n, i, x, dat )
c      where n = input, integer, number of unknowns
c      i = input, integer, equation to be evaluated
c      x(n) = input, double precision, x values
c      dat = input, double precision, data passed into fcn
c      where root only passes data
c      fcn = output, double precision, i-th fcn value
c dat = input, double precision, generic data not used by root but
c      only passed to fcn
c      can be any type, integer, double precision, etc
c mknt = input, integer, maximum iterations to find root
c iknt = output, integer, actual number of iterations used to find root
c
-----

```

```

c      subroutine solvenxn( n, a, b, x )
-----

```

```

c solve system of linear algebraic equations
c
c [A] {x} = {b}
c
c n = input, integer, dimension of arrays
c a(n,n) = input, double precision, matrix

```

c b(n) = input, double precision, RHS vector  
c x(n) = output, double precision, solution vector  
c

---

subroutine triangle( x1, y1, x2, y2, x3, y3, area)

---

c This subroutine calculates the area of the triangle bounded by  
c the points (x1,y1), (x2,y2), and (x3,y3).

c  
c x1,y1,x2,y2,x3,y3 = inputs, double precision, vertices of triangle  
c area = output, double precision, area of triangle  
c

---

subroutine quadrilateral(xysa,quadarea)

---

c this subroutine calculates the area (quadarea) of a quadrilateral

c  
c xysa(2,4) = input, double precision, vertices of quadrilateral  
c quadarea = output, double precision, area of quadrilateral  
c

**C.9 FIND SURFACE ELEVATION utility module (file=findelev.f)**

```
C-----  
  subroutine findelev( xx, yy, ee )  
C-----  
c find elevation of ground surface a UTM coordinates {xx,yy}  
c by Randall D. Manteufel, January 27, 1997  
c  
c xx = input, double precision, UTM Easting coordinate in [m]  
c yy = input, double precision, UTM Northing coordinate in [m]  
c ee = output, double precision, elevation of ground surface  
c      in units of [m] above sea level  
c
```