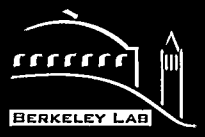
LBNL-44823



ERNEST ORLANDO LAWRENCE
BERKELEY NATIONAL LABORATORY

Statistical Analysis of Liquid Seepage in Partially Saturated, Heterogeneous Fracture Systems

Tai-Sheng Liou

Earth Sciences Division

December 1999

Ph.D. Thesis

DISCLAIMER

LBNL-44823

Statistical Analysis of Liquid Seepage in Partially Saturated, Heterogeneous Fracture Systems

Tai-Sheng Liou
(Ph.D. Thesis)

Department of Civil and Environmental Engineering
University of California, Berkeley

and

Earth Sciences Division
Ernest Orlando Lawrence Berkeley National Laboratory
University of California
Berkeley, California 94720

December 1999

This work was supported by the Director, Office of Science, Office of Biological and Environmental Research, Environmental Sciences Division, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

Abstract

Statistical analysis of liquid seepage in partially saturated, heterogeneous fracture systems

by

Tai-Sheng Liou

Doctor of Philosophy in Civil and Environmental Engineering

University of California, Berkeley

Professor Nicholas Sitar, Chair

Field evidence suggests that water flow in unsaturated fracture systems may occur along fast preferential flow paths. However, conventional macroscale continuum approaches generally predict the downward migration of water as a spatially uniform wetting front subjected to strong imbibition into the partially saturated rock matrix. One possible cause of this discrepancy may be the spatially random geometry of the fracture surfaces and, hence, the irregular fracture aperture. Therefore, a numerical model was developed in this study to investigate the effects of geometric features of natural rock fractures on liquid seepage and solute transport in 2-D planar fractures under isothermal, partially saturated conditions.

The fractures were conceptualized as 2-D heterogeneous porous media that are characterized by their spatially correlated permeability fields. A statistical simulator, which uses a simulated annealing (SA) algorithm, was employed to generate synthetic permeability fields. Hypothesized geometric features that are expected to be relevant for seepage behavior, such as spatially correlated asperity contacts, were considered in the

SA algorithm. Most importantly, a new perturbation mechanism for SA was developed in order to consider specifically the spatial correlation near conditioning asperity contacts. Numerical simulations of fluid flow and solute transport were then performed in these synthetic fractures by the flow simulator TOUGH2, assuming that the effects of matrix permeability, gas phase pressure, capillary/permeability hysteresis, and molecular diffusion can be neglected.

Results of flow simulation showed that liquid seepage in partially saturated fractures is characterized by localized preferential flow, along with bypassing, funneling, and localized ponding. Seepage pattern is dominated by the fraction of asperity contracts, and their shape, size, and spatial correlation. However, the correlation structure of permeability field is less important than the spatial correlation of asperity contacts. A faster breakthrough was observed in fractures subjected to higher normal stress, accompanied with a nonlinearly decreasing trend of the effective permeability. Interestingly, seepage dispersion is generally higher in fractures with intermediate fraction of asperity contacts; but it is lower for small or large fractions of asperity contacts. However, it may become higher if the ponding becomes significant. Transport simulations indicate that tracers bypass dead-end pores and travel along flow paths that have less flow resistance. Accordingly, tracer breakthrough curves generally show more spreading than breakthrough curves for water. Further analyses suggest that the log-normal travel time model generally fails to fit the breakthrough curves for water, but it is a good approximation for breakthrough curves for the tracer.

Statistical Analysis of Liquid Seepage in Partially Saturated, Heterogeneous Fracture Systems

by

Tai-Sheng Liou

Table of contents

List of Figures

Chapter II

Chapter III

Chapter IV

Chapter V

Figure VI

Appendix A

Appendix B

List of Tables

Chapter V

Chapter VI

List of Symbols

(A) Variables in Greek font

Δ	spatial discretization of the flow domain [L]
Ω	a parameter used in van Genuchten's formula for P_{cap} [L^{-1}]
Φ	cumulative distribution function of a normal distribution
Γ	boundary of an element within the flow domain
$\Lambda(\mathbf{h}_r)$	sum of squared differences of $N(\mathbf{h}_r)$ pairs of ζ's
Π	Lagrangian multiplier
Θ	vector of unknown variables in the flow system
α	slope of the log-log plot of the power spectral density of roughness versus frequency
β	kriging weights
χ	a random fluctuation term when considering nugget effect due to sampling error
ε	a constant used to represent the bias of measurement when considering the nugget effect due to sampling error
ϕ	porosity
$\gamma(\mathbf{h})$	semi-variogram at separation distance $\mathbf{h}$
$\gamma_{expected}(\mathbf{h}_r)$	expected semi-variogram of the permeability modifier at separation distance $\mathbf{h}_r$
γ_{UV}	spatial variability between random variables **U** and **V** in terms of semi-variogram
η	maximum allowable ratio of the number of perturbations to κ_{max} when the objective function continues to increase after each perturbation
φ	the dimensionless variable used in Philip's solution, function of θ only
κ_{accept}	maximum number of favored perturbations between two consecutive reductions of temperature
κ_{max}	maximum number of allowable perturbations between two consecutive reductions of temperature correlation length of a random field [L]
λ_k	correlation length of permeability fields [L]
λ_0	correlation length of spatially correlated asperity contacts [L]
λ_p	profile length while measuring surface roughness using profilometers [L]
λ_c	mismatch length scale [L]
μ	viscosity [M/LT]
θ	volumetric moisture content

θ_0	volumetric moisture content at the surface of the one-dimension, semi-infinite soil column
θ_n	initial volumetric moisture content in the one-dimension, semi-infinite soil column
ρ	density [M/L^3]
σ	root-mean-square (rms) roughness of an individual fracture surface [L]
σ_t	standard deviation of travel time [T]
σ_{aw}	surface tension between the wetting and not-wetting phase [M/T^2]
τ	temperature reduction factor, $\tau < 1$
ω	the exponent used in van Genuchten's formulas for k_{rl} and P_{cap}
ξ	range parameter of a semi-variogram [L]
ψ	correlation function
ζ	permeability modifier, $\zeta = k / k_{ref}$
ζ'	permeability modifier after shifting, i.e., $\zeta' = \max(\zeta - \zeta_c, 0)$, for spatially random asperity contacts only
ζ_{avg}	averaged permeability modifier in the entire neighborhood
ζ_c	cutoff permeability modifier, for spatially random asperity contacts only

(B) Variables in small-case Italic font

(n)	neighborhood in indicator kriging
$[i(\mathbf{u})]^*$	estimate of the conditional probability at **u** by indicator kriging
a_i	coefficients in the system of equations of a kriging system
d_i	statistical distance on a scatterplot from a random pair (U_i, V_i) to the perfectly correlated line ($\mathbf{U} = \mathbf{V}$) at 45°
g	gravitational constant [L/T^2]
h	hydraulic head [L]
Δh	head drop across a unit length of fractures [L]
k	fracture permeability [L^2]
k_b	Boltzmann constant
k_{ref}	reference permeability [L^2]
k_{rl}	liquid phase relative permeability
l	size of the neighborhood of an asperity contact
m	mean of a random variable
$m_{\ln\zeta}$	mean of natural logarithm of ζ
N	total number of random pairs (U_i, V_i) on a scatterplot
s	standard deviation of a random variable

s_{D_s}	sample standard deviation of D_s
$s_{\ln(D_s)}$	sample standard deviation of $\ln(D_s)$
s_ζ	standard deviation of the ζ field
$s_{\ln\zeta}$	standard deviation of natural logarithm of ζ
t	time [T]
$<t>$	mean travel time [T]
z	elevation [L]

(C) Variables in small-case font

2b	fracture aperture; hydraulic aperture [L]
b_l	left-hand side fracture aperture in Figure 1.2 [L]
b_r	right-hand side fracture aperture in Figure 1.2 [L]
c	nugget
d	mechanical aperture [L]
$\mathbf{h}$	separation distance between two spatially random variables, $h = \|\mathbf{h}\|$ [L]
$\mathbf{h}_r$	separation distance at r-th lag (or r-th semi-variogram distance) [L]
$\mathbf{n}$	normal vector out of the boundary Γ
nlag	total number of lags at separation distance $\mathbf{h}_r$
n_{xyz}	total number of grid points
p	probability of occurrence of $\mathbf{I}(u)$
p'	acceptance probability of an unfavored perturbation in a thermodynamic system
p_{accept}	acceptance probability of an unfavored perturbation in a SA system
q	mass sink/source [M/L^3T]
r_0	radius of spatially random asperity contacts
u	spatial location [L]

(D) Variables in capital font

A	contact area, or interfacial area [L^2]
$C_I(h)$	covariance of two indicator variables separated by h ($h = \|\mathbf{h}\|$)
C(h)	covariance of two random variables separated by $h = \|\mathbf{h}\|$, also $C_{ij}(\mathbf{u}_i,\mathbf{u}_j)$ or C_{ij} where $h = \|\mathbf{u}_I - \mathbf{u}_j\|$
Cov[U,V]	covariance of random variables U and V
C_q	flow rate constant used in the cubic law [1/LT]
D	diffusivity [L^2/T]

D_f	fractal dimension
D_s	coefficient of seepage dispersion
$\overline{D_s}$	sample mean of D_s
$\overline{\ln(D_s)}$	sample mean of the natural logarithm of D_s
E	system energy of a thermodynamic system
F	mass flux [M/L^{2}T]
I(u)	indicator variable at location **u**
J	Jacobian matrix
K	hydraulic conductivity [L/T]
L	distance from the center of an element to the interface [L]
M	fluid mass [M]
$N(\mathbf{h}_r)$	total number of ζ pairs at separation distance $\mathbf{h}_r$
Nb	total number of grid blocks within the neighborhood of all asperity contacts
NEL	total number of elements
NEQ	total number of equations
NPH	total number of phases
O	objective function in SA
O_{min}	desired objective function at convergence, used to represent the global minimum energy of the SA system
$O_{initial}$	objective function of the initial field
P	pressure [M/LT^2]
ΔP	pressure difference [M/LT^2]
P_{cap}	capillary pressure [M/LT^2]
P_{gas}	gas phase pressure, a constant [M/LT^2]
Q	volumetric flux through fractures [L^3/T]
R	the residual matrix
S_l	liquid saturation
S_w	water saturation
S^*	scaled saturation, $S^* = (S_l - S_{lr})/(1 - S_{lr})$
S_{lr}	residual saturation of the liquid phase
T	temperature parameter in a SA system (or in a thermodynamic system)
T_0	initial temperature in a SA system
V	volume of an element [L^3]
V_l	volume of the liquid phase
V_b	bulk volume of a porous medium

WM	upstream weighting factor
X	mass fraction in a two-component flow system
Y	estimation error between the true spatially random variable and its estimated value from kriging
Z(u)	a general spatially random variable at location **u**

(E) Subscripts

β	fluid phase
l	liquid phase
r	r-th lag
n or m	properties associated with element n or element m
nm	properties at the interface between elements n and m
υ	iteration index

(F) Superscripts

(i)	i-th component in a fluid phase
j	time-stepping index

(G) Operator

∇	gradient operator

Acknowledgments

I would like to express my utmost gratitude to some very special people who have helped me greatly for my graduate study at Berkeley.

First of all, I would like to thank my major advisor, Dr. Nicholas Sitar, for his guidance and patience during the period of my graduate research and the thesis writing. In particular, I would like to give my special thanks to Dr. Karsten Pruess for his professional guidance and support throughout the whole period of my graduate research at the Lawrence Berkeley National Laboratory. I am greatful to Dr. Yoram Rubin for his valuable advice and discussion on my research work. I also thank Dr. James Hunt and Dr. Ted Patzek for their time and efforts in reviewing my dissertation.

I would like to thank the following fellow students at Berkeley, Sheng-Chieh Chang, Wei-Cheng Lo, Julio Garcia, Li-Chiun Lee, Wen-Chin Lee, Yz-Yin Lin, Wei-Hsien Lu, Alex Sun, Chyi-Shan Suen, Ru-Fang Yeh, and Jeong-Seok Yang for their friendship and stimulating discussions throughout the years. I also thank Fu-Chung Wang and Ting-Ruei Shiu for their constant support during my presidency of the Berkeley Association of Taiwanese Students (BATS). Especially, I would like to thank Fei-Wen Chuang for her sincere friendship, and for making my spiritual life more fruitful.

Finally, I would like to express my deepest gratitude and love to my parents and sisters, for their support, unconditional love, and always being by my side.

Chapter I. Introduction

I.1 Motivation

The conventional approach for field-scale analysis of liquid seepage in partially saturated fractured media usually employs macroscale continuum concepts (Peters and Klavetter, 1988). Macroscale volume averaging homogenizes hydrologic properties of the media, such as fracture and matrix permeabilities, and averages spatially variable inputs, such as infiltration rates, applied at the system boundary (Pruess *et al.*, 1999). Consequently, downward water migration in such media is generally modeled as a spatially uniform wetting front, which is subject to strong imbibition into the partially saturated rock matrix (Wang and Narasimhan, 1985, 1993).

However, preferential flow of water and tracer has been observed in the field under saturated or unsaturated conditions. At Rainier Mesa, highly localized flow of water from fractures into drifts was found at depths of several hundred meters beneath the land surface (Thordarson, 1965). At the Stripa mine in Sweden, localized flow paths of water in fractured granite were identified from tracer experiments (Abelin *et al.*,1987), and localized preferential flow was observed in saturated fractured granite (Long *et al.*, 1992). Strongly spatially variable solute concentration and channeling effects were also shown at the Stripa mine (Neretnieks, 1993). At Fran Ridge near Yucca Mountain, lateral migration and preferential flow structures were observed in the densely welded and fractured Topopah Spring tuff (Eaton *et al.*, 1996). Near the Radioactive Waste Management Complex at the Idaho National Engineering and Environmental Laboratory

(INEEL), tracer experiments from the Large-Scale Infiltration Test (LSIT) revealed an irregular distribution of tracer flow, mostly along vertical paths and less so along lateral paths (Wood and Norrell, 1996). Localized preferential flow of water along nonhorizontal fractures has also been observed in laboratory experiments (Nicholl *et al.*; 1994).

Fast preferential flow paths have also been observed at the Yucca Mountain site intended proposed as the Department of Energy (DOE) high-level nuclear waste repository. Geologic units at Yucca Mountain consist primarily of welded and non-welded tuffs, with varying degrees of fracturing in different units. The proposed repository at the Exploratory Studies Facility (ESF) is at approximately 300 m depth. The ESF lies within the unsaturated zone because the water table at Yucca Mountain is approximately 600 m below the land surface. Fracture and fault permeabilities are generally high, on the order of 1 – 10 darcies and 10 – 100 darcies, respectively (Ahlers *et al.*, 1996). In contrast, the matrix permeabilities are on the order of 1 – 10 microdarcies (Flint, 1997). The contrast of permeability in fractures and the rock matrix suggests that most of the flow must preferentially go through fractures and major faults. For example, field experiments using environmental isotopes found elevated levels of ^{36}Cl at several locations in the ESF (Fabryka-Martin *et al.*, 1996). If the effect of imbibition into the partially saturated rock matrix were significant, the travel time of water from the land surface to the water table would require thousands of years and the corresponding water velocity was roughly estimated to be on the order of 50 mm/year (Pruess *et al.*, 1999). However, field experiment data (Fabryka-Martin *et al.*,1996) suggest that water seepage through Yucca Mountain occurs with velocities on the order of 10 m/year or faster

(Pruess *et al.*, 1999). In addition, calcite-coating data show that deposition is found mostly within fractures and not within the matrix. Hence, the diffusion of water into the rock matrix is very low. In summary, all evidence suggests that in semi-arid environments water is able to migrate downward rather rapidly along localized preferential flow paths through fracture networks in partially saturated rocks, without being imbibed into the rock matrix.

I.2 Approach

Unsaturated flow in naturally fractured rocks is generally a multi-phase, non-isothermal flow that occurs in a three-dimensional fracture network. A complicated model is generally needed to model the actual fluid flow and transport in natural fractures. However, the primary interest of this study is to understand the effect of geometric features of natural rock fractures on gravity-driven liquid seepage in fractures under isothermal, partially saturated conditions. Thus, the following assumptions have been made to simplify the modeling:

a. Impact of the gas phase on seepage is neglected by assuming that gas phase pressure is a constant. Namely, effects associated with the gas phase fluid, such as the dramatic change of hydraulic characteristics of porous media by trapped air (Faybishenko, 1995), are not considered in this study. For systems with small capillary numbers, it is reasonable to ignore gas phase pressure. By making this assumption, the total number of phases in the system is reduced by one and only the balance equation of the wetting phase fluid (e.g., water) needs to be considered for two-phase problems. Consequently, the remaining

unknown variables will be either liquid phase saturation for one-component flow systems, or liquid saturation and mass fraction of the second component liquid for two-component flow systems. Although the gas phase is assumed to be stagnant, phase interference is still considered by specifying the relative permeability of the aqueous phase.

b. Matrix permeability and the interaction between the fractures and the surrounding rock matrix is negligible. This assumption is based on the following considerations. First, field data show that matrix permeability for some rock types, e.g., welded tuff, is orders of magnitude smaller than fracture permeability. Second, the effect of matrix permeability on seepage evolves at a much longer time scale (months to years) than the effect of fracture permeability (usually hours). For solute transport, molecular diffusion may dominate the interaction between fractures and the rock matrix, which is also a slow process. Thus, for shorter time scale simulations, the effect of the rock matrix can be neglected.

c. Hysteresis effects of capillary pressure as well as permeability are neglected. Hysteresis of capillary pressure occurs when fractures are subject to repeated wetting and drying cycles. Permeability hysteresis occurs when fractures are undergoing repeated loading/unloading cycles. Since most simulations in this study consider single wetting events without loading/unloading cycles, hysteresis effects are not important.

In addition to the above assumptions, this study focuses on studying seepage behavior in planar two-dimensional fractures that are conceptualized as 2-D heterogeneous porous media. Approximation of 3-D fracture networks as 2-D heterogeneous porous media is only applicable to small fractures in hard rocks of low permeability, such as welded tuffs, graywacke, mudstones, granite, and some fractured basalts. It would not be applicable to larger fractures with 3-D void space, or to small fractures in rocks with significant matrix permeability, such as non-welded tuffs and sandstones. Of course, 3-D flow effects cannot be adequately modeled in a 2-D framework. However, such conceptualization is believed to be sufficient for the purpose of fundamental understanding of flow and transport in 3-D fracture networks. An immediate advantage of using such conceptualization is that the effective properties of porous media, such as relative permeability and capillary pressure, can be substituted for fractures. Indeed, the similarity between porous media and fractures in terms of relative permeability and capillary pressure has been verified experimentally (Persoff and Pruess, 1995).

Fluid flow in single fractures can be conveniently analyzed by a continuum approach. However, important flow mechanisms in partially-saturated fractured rock usually operate at microscales such that the macroscale volume-averaged parameters or system of equations may not capture all the significant mechanisms. For example, macroscale continuum approaches generally fail to predict preferential flow observed in partially saturated fractured media such as Yucca Mountain. Furthermore, predictions

based on macroscale continuum approaches may become totally meaningless if a great volume of the flow system is bypassed due to fracture heterogeneities.

The following approach is then proposed to overcome the conceptual difficulty of macroscale continuum approaches for modeling fluid flow and transport in variably saturated fractured media. First, 3-D fracture systems are approximated as 2-D planar fractures that are conceptualized as 2-D heterogeneous porous media. Volume-averaged parameters for porous media such as porosity, permeability and capillary pressure are all expected to show spatial variability. However, this study focuses on permeability heterogeneity in the fracture plane. Heterogeneous permeability fields generated with a statistical simulator at a high spatial resolution are used to characterize the porous media. Then, a volume-averaged Richards' equation is employed to model the flow behavior in the equivalent porous media. The difference between this approach and conventional continuum approaches is that fracture heterogeneity (permeability) is explicitly incorporated into the Richards' equation. Thus, it is expected to capture important seepage mechanisms that may be overlooked by continuum approaches, such as flow bypassing and channeling.

I.3 Objectives

Based on to the evidence of fast preferential flow at sites with thick unsaturated fractured zones, several researchers have proposed to conceptualize unsaturated flow in heterogeneous fractured media as a stochastic distribution of localized seeps (Gauthier *et al.*, 1992; Gauthier, 1994), i.e., the Weeps model. Although oversimplified, the Weeps model is important because it implies the relationship between the fast preferential flow

and geometric features of natural fractures. Accordingly, a mechanistic process model combined with statistically characterized fracture heterogeneity is used herein to evaluate flow and transport behavior of natural fractures. The appropriateness of the current approach is judged on the basis of how relevant the assumptions of significant geometric properties of fractures are to field observations. However, our ability to directly obtain geometric characteristics of fracture void spaces from field observations is very limited. Only input into and output from the flow system at the boundaries can be obtained from field observations, which can be linked only implicitly to the assumptions for synthetic rock fractures. Therefore, the objectives of this study are

(1) to evaluate what geometric features of natural rock fractures determine gravity-driven liquid seepage in partially saturated conditions.

(2) to provide building blocks for a theory of liquid seepage in partially saturated fracture systems, formulated in terms of statistical properties of ensemble of seeps.

(3) to develop guidance for observing, sampling and testing in partially saturated fractures systems, in order to obtain meaningful field characterization.

Chapter II. Background – Fracture properties and fracture flow

II.1 Surface properties of natural fractures

Natural fractures are characterized by their spatially varying aperture geometry and heterogeneous permeability. These properties are the result of the spatial variability and correlation of the rough surface of fractures. Accordingly, the general approach for characterizing natural fractures is to conceptualize them as two rough surfaces that contact each other at discrete points, and are spatially correlated with each other at different scales (Brown, 1995).

The topography (roughness) of fracture surfaces determines not only the mechanical but the hydraulic/transport properties of fractures (Glover *et al.*, 1998a; Brown, 1987ab, 1989; Pyrak-Nolte *et al.*, 1987; Brown and Scholz, 1985b; Kranz *et al.*, 1979). While the shape, size, and number of contact points between fracture surfaces control mechanical properties of rock, geometrical properties of fracture surfaces control fluid flow in fractured rocks. Thus, geometric properties of fracture surfaces as well as the resulting fracture permeability and aperture are important factors for understanding fluid flow and solute transport in unsaturated fractures.

II.1.1 Fracture permeability

Fracture permeability can be theoretically defined by the parallel plate model (Witherspoon *et al.*, 1980). This model has been traditionally used to study the steady state, single-phase, isothermal and saturated flow of incompressible fluids in single

fractures. In this model, naturally rough fracture surfaces are idealized as two smooth, parallel plates that are separated by a constant aperture (2b), see Figure II.1.

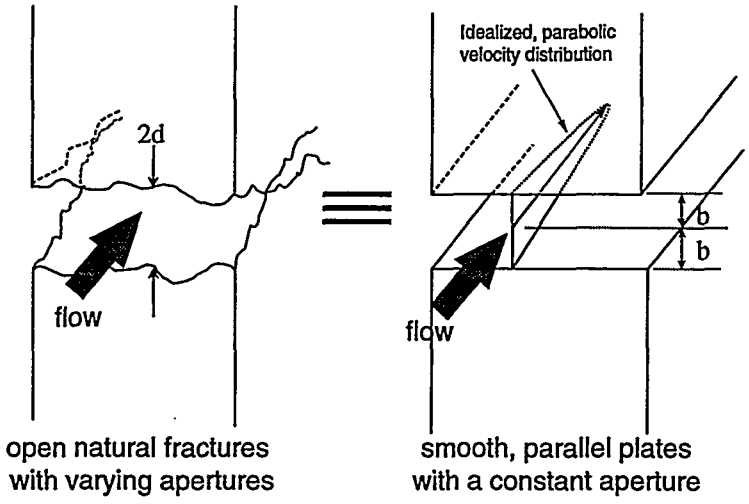


Figure II.1 Schematic representation of the parallel plate model.

Analytic analyses such as Bear (1972) show that fracture permeability (k) has the following relationship to fracture aperture

$$k = \frac{(2b)^2}{12} \tag{2.1}$$

Equivalently, fracture transmissivity (T) is found to be proportional to the cube of the fracture aperture, i.e., $T \approx (2b)^3$. In addition, fluid flux per unit drop in head can be developed from Darcy's law, which may be written in a simplified form as

$$\frac{Q}{\Delta h} = C_q (2b)^3 \tag{2.2}$$

where Q is the volumetric flow rate across the fracture, Δh is the head drop, and C_q is a constant depending on flow geometry and fluid properties (Witherspoon *et al.*, 1980).

Equation (2.2) has been referred to as the "cubic law" in the literature. What is also predicted from the parallel plate model is that the flow field within the fracture has a symmetric parabolic velocity distribution, see Figure II.1. This well-developed velocity distribution has known to be contrary to field observations. Accordingly, the lack of consideration of the spatially varying roughness and the spatial correlation of aperture is the key to the failure of the parallel plate model to predict the real flow field in natural fractures (Wang and Narasimhan, 1988).

Indeed, experimental as well as theoretical studies have shown that permeability of single, natural fractures is a complicated parameter depending on several factors. Kranz *et al.* (1979) found that the higher the surface roughness (the mean asperity height) of jointed fractures, the slower the decline of permeability with increasing effective stress (the difference between the external confining pressure and the internal fluid pressure). In addition, they found that fracture permeability decreases nonlinearly with increasing effective stress and increasing sample size. This trend indicates that effective permeability approaches asymptotically to the lower limit at zero for fractures subjected to increasing normal stress. It also implies that residual flow may exist even if the apparent fracture aperture is essentially zero. The existence of residual flow in fractures at high normal stress is consistent with experiments previously reported by Iwai (1976) and Raven and Gale (1985). Walsh (1981) attributed the decrease of permeability with increasing normal stress to the decrease of aperture, increase of contact points, and increase of tortuosity of flow paths. In addition, fracture permeability measured in the laboratory exhibits a significant hysteresis effect during loading and unloading cycles (Raven and Gale, 1985;

Kranz *et al.*, 1979). This is attributed to plastic deformation caused by crushing of asperities in jointed fractures, or irrecoverable damage in intact rocks (Kranz *et al.*, 1979). Zimmerman *et al.* (1992) found that, regardless of the geometry of asperities, fracture permeability decreases nonlinearly with increasing fraction of contact area. Experimental data show that flow path tortuosity of natural fractures increases with normal stress (Pyrak-Nolte *et al.*, 1987). While Walsh (1981) suggested that tortuosity is not an important parameter in estimating the flow rate through fractures, Tsang (1984) showed that neglecting tortuosity effects may result in one to two orders of magnitude error in computing the flow rate. Moreover, fracture permeability also depends on contact area.

In general, factors controlling fracture permeability include fracture aperture, sample size, surface roughness, contact area, tortuosity, normal stress, stress history, scale of measurement and rock type. Permeability measured in the laboratory is generally several orders of magnitude smaller than that in the field (Brace, 1980). Furthermore, permeability of jointed rock is much greater than that of intact rock (Kranz *et al.*, 1979), implying that fluid flow is confined essentially to joints and fractures in the rock.

II.1.2 Fracture aperture

An important aspect of modeling flow and transport in natural fractures is the ability to describe their spatial variability of the aperture geometry. Although fracture aperture can be inferred from surface roughness, it is generally difficult to measure surface roughness in fractures in-situ, especially on a large scale. Another difficulty is posed by multiple definitions of fracture aperture and orders of magnitude differences between various definitions. The most commonly used definitions are "hydraulic

aperture" and "mechanical aperture". The hydraulic aperture, 2b, is defined as the constant opening between the two smooth surfaces in the parallel plate model. Recall that it is related to the intrinsic fracture permeability (k) by $k = (2b)^2/12$. The mechanical aperture, d, is defined as the mean separation distance between two fracture surfaces that are held parallel to each other. It is not a constant but varies non-linearly with normal stress (Raven and Gale, 1985). In addition, the mechanical aperture depends on the details of fracture surface topography as well as the elastic properties of fractures (Brown and Scholz, 1986). Another definition of aperture is the "volumetric aperture" (Abelin *et al.*, 1987), i.e., the fracture void volume per unit fracture surface area. It is often orders of magnitude larger than the hydraulic aperture and the mechanical aperture (Abelin *et al.*, 1987). In addition, hydraulic aperture may underestimate the mean residence time for the water (Abelin *et al.*, 1987). This implies that tracer breakthrough curves predicted from the hydraulic aperture may have earlier arrival of the peak concentration than that predicted from the volumetric aperture.

Much experimental research has been done in the last decade to explore the void space geometry between fracture surfaces. Most experiments used fluid injection and image processing methods to study this property of natural fractures. Mercury porosimetry method is one of the methods used (Myer *et al.*, 1993). However, the wood's metal injection method is more popular than the mercury porosimetry method. The advantage of Wood's metal is that it can yield the actual metal casts of the voids for the same fracture in experiments at different stresses (Pyrak-Nolte *et al.*, 1987). For example, the micrographs in Figure II.2 were obtained by Pyrak-Nolte *et al.* (1987) by the Wood's

metal injection method to characterize the void geometry for a natural fracture in granite, which is subjected to increasing normal stress from 3 MPa, 33 Mpa, to 85 MPa.

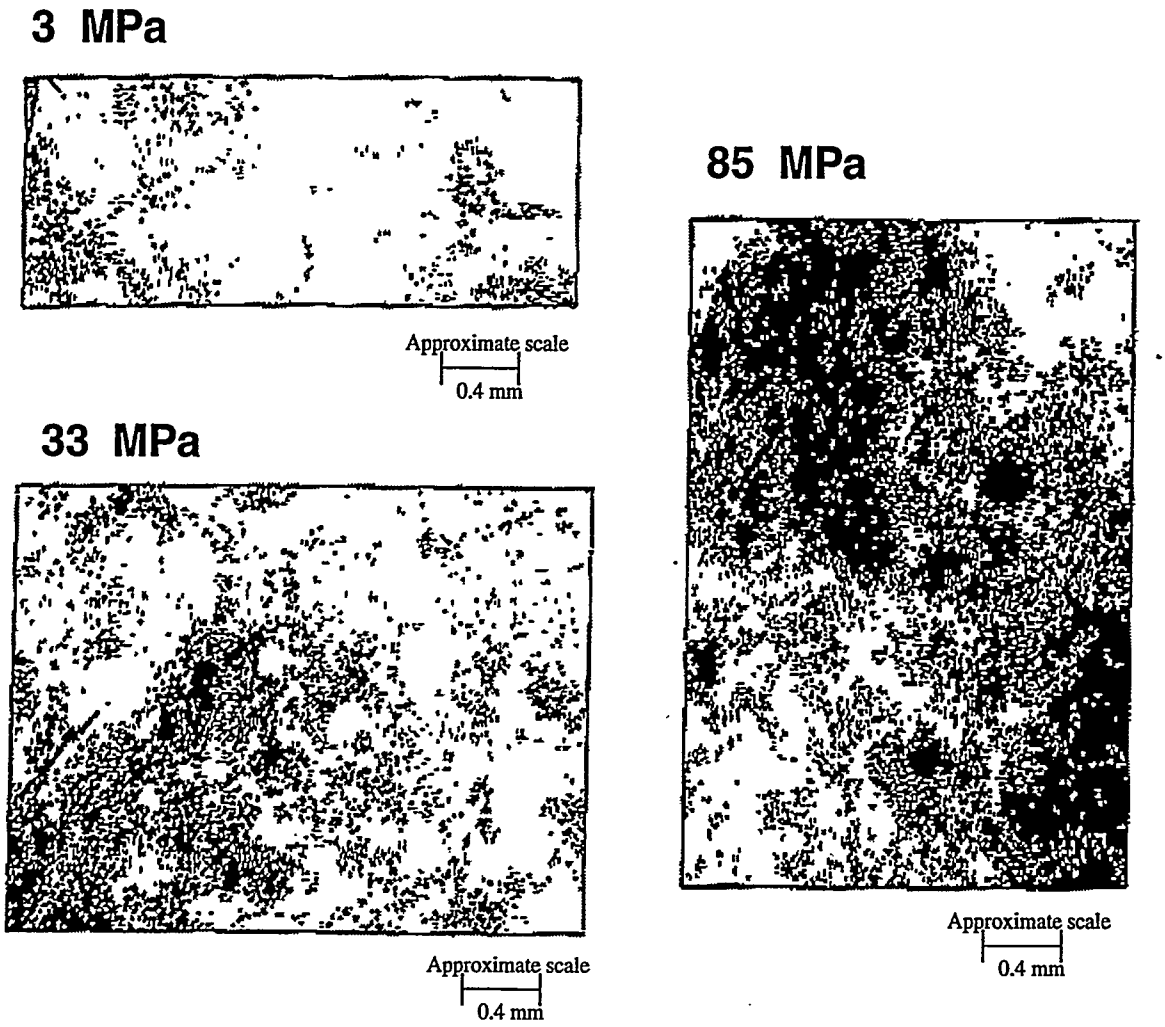


Figure II.2 Change of void geometry in natural Stripa granites with respect to normal stress, measured by Pyrak-Nolte *et al.* (1987) using the Wood's metal injection method. Asperity contacts (or inaccessible pore space) appear as black regions, while regions penetrated by Wood's metal are white. Note that these micrographs were take from different portions of the same sample.

Figure II.2 shows that contact areas generally increase nonlinearly with the normal stress (Pyrak-Nolte *et al.*, 1987). In addition, flow paths in the fracture plane becomes

more tortuous as the normal stress increases. Since contact areas and voids are spatially correlated, a contact point (or a void site) is likely to be surrounded by other points of contact (or other void sites) (Pyrak-Nolte *et al.*, 1990). This spatial correlation structure could be numerically approximated by an exponential function (Coakley *et al.*, 1987).

Based on direct measurement data, several researchers also found that roughness of natural fracture surfaces can be described by a characteristic length scale (Brown and Scholz, 1986; Brown *et al.*, 1986). The asperities are correlated below this scale and uncorrelated above. Hence, this correlation length scale is called the mismatch length scale, λ_c. It may also be used as the cutoff wavelength for the scaling law of fracture aperture (Brown, 1995). Namely, fracture aperture is scale dependent only if the wavelength of roughness is smaller than λ_c.

II.1.3 Surface roughness

Surface roughness is a small-scale characteristic of natural fracture surfaces. It can be measured in the field as well as in the laboratory by a profilometer (Brown *et al.*, 1986; Brown and Scholz, 1985ab, 1986; Power *et al.*, 1987; Glover *et al.*, 1998b). After comparing the roughness of various natural rock surfaces, Brown and Scholz (1985a) concluded that fracture surfaces are fractal in nature. Thus, the surface profile of an individual fracture surface can be decomposed into a series of sinusoidal Fourier waves, each of them have a wavelength, amplitude, and phase. Surface roughness depends on sample size and the scale of observation (Brown and Scholz, 1985). For example, Brown (1995) showed that the scaling law for an individual fracture surface can be written as $\sigma \sim$

$\lambda_p^{(\alpha-1)/2}$, where σ is the root-mean-square (rms) roughness (or the standard deviation of the height of a fracture surface), λ_p is the wavelength of a sinusoidal Fourier wave, and α is the slope of the log-log plot of the power spectral density of roughness versus frequency. Surface roughness is an important in controlling laminar flow through natural fractures in theoretical, numerical as well as experimental studies (Walsh, 1981; Brown,1987a; Pyrak-Nolte *et al.*, 1987, Pyrak-Nolte *et al.*, 1988; Brown, 1989; Zimmerman *et al.*, 1992).

The void space geometry in fractured rocks may span multiple scales. It may range from a small scale (roughness), intermediate scale (asperity contacts, fracture intersections and terminations) to large scale (network connectivity). This property is due to the small-scale variability of an individual surface and the spatial correlation of the contacting fracture surfaces. It is then expected that fluid particles will take a tortuous flow path when moving through a real fracture.

Brown (1995) suggested that only a few parameters are needed to exhaustively characterize natural fracture surfaces. These parameters are the rms roughness (σ), fractal dimension (D_f), and the mismatch length scale (λ_c). Fractal dimension is also used to measure the scaling of fracture surfaces, i.e., $\sigma \sim \lambda_p^{3-D_f}$ with $\alpha = 7 - 2D_f$ (Brown, 1995, 1987a). Recall that the mismatch length scale is also defined as the cutoff length-scale specifying the correlation/un-correlation of fracture surfaces. In reality, however, fracture surfaces may vary over a broad range of wavelengths (or inversely, frequencies). Thus, the unique cutoff mismatch length-scale employed in Brown's model does not seem to be

adequate for modeling real fractures. This difficulty has been overcome by Glover *et al.* (1998) by taking into account the smoothly varying degree of mismatch in natural fractures. Of course, fractal models are not the only approach of characterizing natural fractures. Recent success of employing a statistical simulator other than fractal models for generating synthetic replica of fractured rocks has been reported in the literature. For example, Pruess and Antenuz (1995) used the turning band method (TBM) to generate synthetic fractures in terms of fracture permeability.

II.2 Limitations and applicability of the cubic law

Numerous experimental as well as theoretical studies have been done to investigate the applicability of the cubic law to natural fractures. In general, the cubic law appears to be applicable to fluid flow through loosely mated and open fractures, as well as to fractures with high correlation between fracture surfaces (Nolte *et al.*, 1989). Experimental works by Witherspoon *et al.* (1980) and Iwai (1976) indicated that the cubic law is generally valid independent of the rock type. In addition, numerical simulations by Brown (1987) showed that the actual flow rate asymptotically approaches that predicted by the cubic law as the ratio of fracture aperture to rms roughness increases.

However, the cubic law generally tends to overestimate the actual flow rate in natural fractures. Tsang and Witherspoon (1981) found that the flow rate predicted by the cubic law has to be reduced if surface roughness is taken into account. Brown (1987) found that the actual flow rate is only 40% - 60% of that predicted by the cubic law if the ratio of fracture aperture to rms roughness is one; but is increased to 70% - 90% if this

ratio is between 2 to 4.24. Nolte *et al.* (1989) established an empirical power law of flow rate to mechanical aperture based on flow experiments performed on Stripa granite. They found that the volumetric flow rate is proportional to aperture raised to a power greater than 3 and close to 8, suggesting that the cubic law may not adequately describe natural fractures. In addition, permeability predicted from the cubic law was found to be orders of magnitude higher than that measured from experiments (Kranz *et al.*,1979; Raven and Gale, 1985).

In general, the cubic law is not applicable to rough fractures under high normal stresses. As contact areas in fractures increase with increasing normal stress (Nolte *et al.*, 1989), the actual flow paths become more tortuous and channeled (Raven and Gale, 1985; Brown, 1987b; Glover *et al.*, 1998b; Pyrak-Nolte *et al.*, 1987). Thus, Pyrak-Nolte *et al.* (1988) found that the flow rate predicted by the cubic law for fractures at high normal stress significantly differs from measured data. They suggested that this difference may be a consequence of the dominating influence of a critical neck (the point of smallest aperture along the path of highest aperture) on flow through the fracture.

It is evident that natural fractures should be characterized by a spatially varying aperture distribution. From a numerical point of view, some researchers, e.g., Pruess and Tsang (1990), adopted the approach that fracture surfaces can be locally approximated as two parallel plates separated with a constant aperture. In addition, the cubic law is assumed to be locally valid within that pore space. However, several aspects need to be considered before adopting this approach.

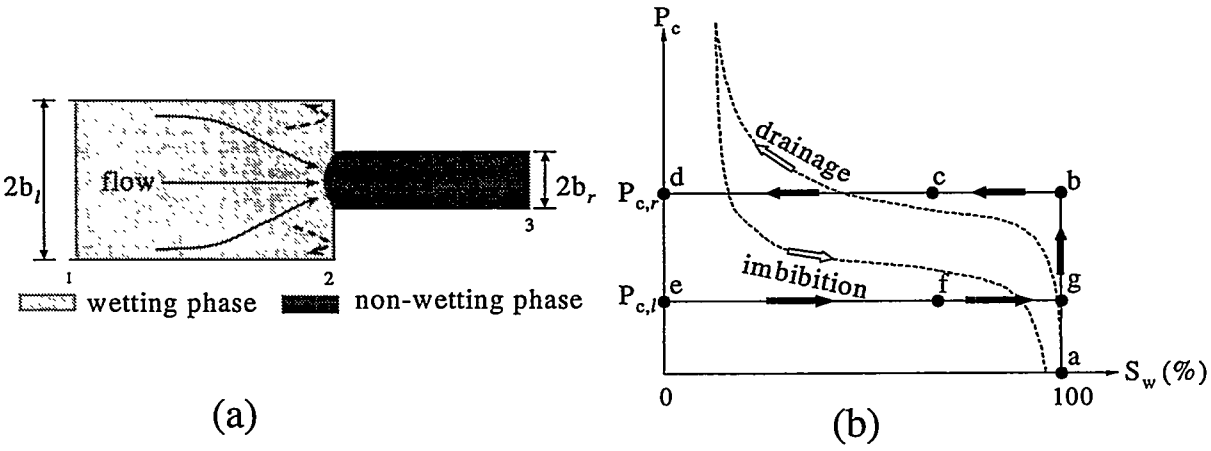


Figure II.3 (a) Schematic diagram of natural fractures idealized as smooth and parallel plates, (b) Sketch of the capillary pressure as a function of the saturation of the wetting phase fluid, for both the parallel plate pore model and a real porous medium.

Consider two pores idealized as parallel plates as shown schematically in Figure II.3(a). Assuming that the pores are initially filled with a wetting phase fluid. Also, assuming that the system is connected to the right with the non-wetting phase, and to the left the wetting phase fluid. The drainage process can be initiated if the pressure difference between the wetting and non-wetting phase fluids is large enough to overcome the capillary pressure $P_{c,r}$ ($= 2\sigma_{aw}/b_r$). Thus, the initial drainage curve will follow line ab in Figure II.3(b). Subsequently, the system is drained from location 3 to location 2 in Figure II.3(a), corresponding to line bc in Figure II.3(b). Since the capillary pressure needed to drain the larger pore ($P_{c,l} = 2\ \sigma_{aw}/b_l$) is smaller than $P_{c,r}$, the non-wetting phase fluid will completely penetrate the larger pore as soon as the interface arrives at location 2 in Figure II.3(a). This is reflected by line cd in Figure II.3(b). After the wetting phase is completely drained from the larger pore, an equilibrium capillary pressure ($P_{c,l}$) is reached, which is indicated by the dashed interface at the left-hand side of Figure II.3(a).

With the help of capillary pressure, the wetting phase fluid can be imbibed into the larger pore if the pressure of the wetting phase fluid is slightly increased. Hence, the initial imbibition process will follow line ef in Figure II.3(b), and the interface will advance from location (1) to location (2). Since the capillary pressure at the pore throat (location (2) in Figure II.3(a)) is larger than $P_{c,l}$, the wetting phase fluid will be sucked into the smaller pore as soon as the interface reaches location (2) in Figure II.3(a). Thus, the entire imbibition process follows line efg in Figure II.3(b).

In reality, however, the drainage/imbibition processes for a real porous medium would follow the dashed curve in Figure II.3(b). Moreover, if the flow velocity is large, flow dynamics may become dominant at the pore throat where significant change of surface curvature occurs. Counter-current eddies as shown by the dashed arrows in Figure II.3(a) may develop due to the large flow velocity and may result in over-estimation of flow rate calculated based on the parallel-plate model. Therefore, neglecting the small-scale wall roughness of natural fractures may result in unrealistic approximation of the real flow field in fractures. This is especially true for field scale applications because idealizing field scale fractures as parallel plates certainly suffers from the difficulty of capturing the small scale surface roughness. Overall, this overview shows that there is a need to develop fracture flow models which adequately and realistically describe the spatial variability of the fracture aperture. This is the approach pursued herein.

Chapter III. Statistic Modeling of Fractures with Spatially Varying Roughness

III.1 Introduction

Modeling of flow and transport in fractured rocks or porous media very often faces the problem of incomplete information about the heterogeneity of the media. Thus, stochastic simulation has become a common tool for characterizing and visualizing medium heterogeneity based on incomplete information. To reduce the uncertainty of predicting heterogeneity, it is favored to incorporate field data from a variety of sources into a simulator, e.g., borehole logs (Johnson and Dreiss, 1989), seismic data (Copty and Rubin, 1995), and tracer concentration data (Dagan *et al.*, 1997). Such simulations not only try to reduce the uncertainty of characterization but honor the sample data. Unfortunately, no stochastic simulators can perfectly reproduce the reality of the field and most simulators cannot make use of all available information. Moreover, some simulators are restricted to Gaussian random fields only, e.g., the turning bands method (TBM) (Mantoglou and Wilson, 1982; Tompson *et al.*, 1989), COVAR (Williams and El-Kadi, 1986; Abdel-Salam and Chrysikopoulos, 1996), and spectral methods (Shinozuka and Jan, 1972). However, discrete or combinatorial optimization methods, such as simulated annealing (SA), have shown great promise in their applicability to various random fields and their ability to incorporate data from various sources into their models by formulating a suitable objective function (Datta-Gupta *et al.*, 1995; Deutsch and Journel, 1994). In order to model fracture characteristics, the stochastic simulator needs to be able to model

the following elements: (1) the presence of asperity contacts, (2) a gradual change towards larger apertures away from the asperities, (3) fracture wall roughness, and (4) spatial correlation structure of fracture aperture (Pruess and Antunez, 1995). Simulated annealing (SA) is chosen in this study as the numerical simulator to characterize fracture heterogeneity because it very well satisfies these objectives, as discussed next.

III.2 Simulated annealing algorithm

Simulated annealing (SA) is an algorithm originally developed for combinatorial optimization, i.e., optimizing a system with discrete variables. The heart of SA is an analogy with a thermodynamic system i.e., the physical process of annealing materials such as semiconductors and metals (Deutsch and Journel, 1994). It is effective for large-scale systems with discrete variables (Kirkpatrick *et al.*, 1983). However, it can also be applied for optimizing a system with continuous variables (Press *et al.*, 1986). It has been successfully applied in a great variety of fields involving computer design (Kirkpatrick *et al.*, 1983), nonlinear geophysical inversion (Sen and Stoffa, 1991), and stochastic reservoir modeling (Deutsch and Journel, 1994). In hydrology, SA was first employed by Dougherty and Marryott (1991) for finding an optimal groundwater management strategy. Several computer codes of SA are available in the literature. The computer code used in this study is updated from the subroutine SASIM in the software library GSLIB (Deutsch and Journel, 1992).

To be able to "anneal" the numerical system in a way similar to annealing a thermodynamic system, a SA algorithm must contain the following four components: (1)

an initial state, (2) an objective function that is to be minimized, (3) perturbation mechanism, and (4) annealing schedule. (Deutsch and Cockerham, 1994), which will be explained in detail in the following sections.

III.2.1 Initial field

The initial field for SA can be a purely random field or a field that already shares certain spatial features of the desired random field (Datta-Gupta *et al.*, 1995). For a purely random field, many perturbations may be needed to reach the optimal state. However, the performance of the SA algorithm may be improved if the initial state already has some spatial features (Johnson *et al.*, 1989).

As mentioned in Chapter I, fracture permeability (k) is used to characterize heterogeneous fractures. For convenience, permeability is scaled by a constant reference permeability, k_{ref}. The scaled permeability is called the permeability modifier and is symbolized as ζ, i.e., $k = k_{ref} \times \zeta$. A reasonable value of the reference permeability for field-scale fractures, e.g., welded tuff, may be 10^{-9} m^2 (1000 d). Asperity contacts, i.e., regions with zero permeability, are simply modeled as $\zeta = 0$.

All the initial states in this study are generated in the following two steps. First, the conditioning asperity contacts are generated by a pre-processor. All conditioning data are asperity contacts; however, not all asperity contacts are conditioning data, see section III.2.1.2. Two different pre-processors are used for generating the conditioning asperity contacts. The difference between these pre-processors is their ability to consider the spatially correlation of asperity contacts. Second, the un-conditioned grid blocks are filled

with data drawn from a known probability distribution. The probability distribution used in this study was assumed to be log-normal (see Eq(3.1)), but other distributions may also be possible.

$$f(\zeta) = \frac{1}{\sqrt{2\pi}\, s_{\ln\zeta}\, \zeta} \exp\left[-\frac{(\ln\zeta - m_{\ln\zeta})^2}{2 s_{\ln\zeta}^2} \right] \tag{3.1}$$

Mean ($m_{\ln\zeta}$) and standard deviation ($s_{\ln\zeta}$) of the log-normal distribution, were chosen as 1.0 and 1.5, respectively. Note that SA does not require that the random field be Gaussian (Dutta-Gupta *et al.*, 1995). In contrast, the initial field can be drawn from a variety of sources. For example, field sampled data (the conditioning data) plus random values drawn from a known distribution (the un-conditioned data) may be used.

Asperity contacts with and without spatial correlation are considered in this study, which are referred to as spatially correlated and spatially random asperity contacts, respectively. Indicator simulation and Boolean simulation are the corresponding pre-processors for generating these types of asperity contacts. Both pre-processors can be found in GSLIB (Deutsch and Journel, 1992).

III.2.1.1 Spatially correlated asperity contacts

Since fracture surfaces are spatially correlated to each other, asperity contacts (regions where two fracture surfaces contact each other) are also expected to be spatially correlated. Accordingly, the micrographs shown in Figure II.2 illustrate that asperity contacts (the black regions) are clustered with a specific spatial correlation. Recall that this spatial correlation can be approximated by an exponential function (Coakley *et al.*,

1987). Moreover, these micrographs imply that the void space in a fracture plane can be approximated by a binary process. Namely, the void space at a specific location in the fracture plane is either closed (asperity contact) or open (aperture). Hydrologic parameters in some porous media may also have this binary property, such as hydraulic conductivity in sand-shale or sand-clay formations in fractured rock (Rubin, 1995), or effective permeability in sand-shale formations (Desbarats, 1987, 1990). Statistically, a binary process can be described by an indicator function (Journel, 1983). Therefore, indicator simulation (Deutsch and Journel, 1992) used in geostatistics is employed herein to simulate spatially correlated asperity contacts in natural fractures.

III.2.1.1(a) Indicator simulation

Indicator simulation is a linear regression algorithm which sequentially updates the estimation of a spatially random variable with conditioning information collected from a suitable neighborhood (Deutsch and Journel, 1992). The size of neighborhood will be discussed at the end of III.2.1.1(c). Indicator simulation is ideally suitable for simulating binary variables, for example, asperity contacts and void space in a fracture plane.

A binary, spatially random variable $\mathbf{Z}(\mathbf{u})$, such as the aperture field in natural fractures, can be defined in terms of an indicator function $\mathbf{I}(\mathbf{u})$. Hereafter in this chapter, a bold capital letter refers to a spatially random variable, while a italic capital letter is its realization. The indicator function is a spatially random function (SRF), and can be defined as

$$\mathbf{I}(\mathbf{u}) = \begin{cases} 1\ , \text{ probability} = p(\mathbf{u}) \text{ for } \mathbf{u} \in s \\ 0\ , \text{ probability} = 1 - p(\mathbf{u}) \text{ for } \mathbf{u} \notin s \end{cases}, s \equiv \text{asperity contacts} \tag{3.2}$$

where $\mathbf{u}$ is the spatial coordinate. Likewise, we can define another indicator variable ($\mathbf{I}'(\mathbf{u})$) for the counterpart of the binary variable, e.g., void space in a fracture plane. That is,

$$\mathbf{I}'(\mathbf{u}) = \begin{cases} 1,\ \text{probability} = 1 - p(\mathbf{u}) \text{ for } \mathbf{u} \in s' \\ 0,\ \text{probability} = p(\mathbf{u}) \text{ for } \mathbf{u} \notin s' \end{cases}, s' \equiv \text{void space} \tag{3.3}$$

The expectation of an indicator variable can be derived as

$$E[\mathbf{I}(\mathbf{u})] = 1 \cdot p(\mathbf{u}) + 0 \cdot (1 - p(\mathbf{u})) = p(\mathbf{u}) \tag{3.4}$$

i.e., the expected value of an indicator variable is its probability of occurrence. Similarly, the expected value of $\mathbf{I}'(\mathbf{u})$ is 1-p($\mathbf{u}$). The variance of an indicator variable is

$$\mathrm{Var}[\mathbf{I}(\mathbf{u})] = E\left[(\mathbf{I}(\mathbf{u}) - E[\mathbf{I}(\mathbf{u})])^2\right] = E\left[(\mathbf{I}(\mathbf{u}))^2\right] - E[\mathbf{I}(\mathbf{u})]^2 = p(\mathbf{u}) \cdot (1 - p(\mathbf{u})) \tag{3.5}$$

Similarly, variance of the counterpart indicator variable ($\mathbf{I}'(\mathbf{u})$) is also p($\mathbf{u}$)(1- p($\mathbf{u}$)). The covariance of two indicator random variables separated by a distance $\mathbf{h}$ is

$$\begin{aligned} \mathrm{Cov}[\mathbf{I}(\mathbf{u}), \mathbf{I}(\mathbf{u}+\mathbf{h})] &= E[\mathbf{I}(\mathbf{u}) \cdot \mathbf{I}(\mathbf{u}+\mathbf{h})] - p(\mathbf{u}) \cdot p(\mathbf{u}+\mathbf{h}) \\ &= 1 \cdot 1 \cdot \mathrm{Prob}\{I(\mathbf{u}) = 1, I(\mathbf{u}+\mathbf{h}) = 1\} + 1 \cdot 0 \cdot \mathrm{Prob}\{I(\mathbf{u}) = 1, I(\mathbf{u}+\mathbf{h}) = 0\} \\ &\quad + 0 \cdot 1 \cdot \mathrm{Prob}\{I(\mathbf{u}) = 0, I(\mathbf{u}+\mathbf{h}) = 1\} + 0 \cdot 0 \cdot \mathrm{Prob}\{I(\mathbf{u}) = 0, I(\mathbf{u}+\mathbf{h}) = 0\} \\ &\quad - p(\mathbf{u}) \cdot p(\mathbf{u}+\mathbf{h}) \\ &= \mathrm{Prob}\{I(\mathbf{u}) = 1, I(\mathbf{u}+\mathbf{h}) = 1\} - p(\mathbf{u}) \cdot p(\mathbf{u}+\mathbf{h}) \end{aligned} \tag{3.6}$$

An SRF is stationary if its cumulative distribution function (CDF) is invariant to spatial translation. Therefore, the mean and variance for a stationary SRF are constants,

and the covariance depends only on the magnitude of the separation distance (|**h**| or h). If the indicator random variable I(**u**) is a stationary SRF, its mean, variance and covariance can be rewritten as

$$\begin{aligned} E[\mathbf{I(u)}] &= p \\ Var[\mathbf{I(u)}] &= p(1-p) \\ Cov[\mathbf{I(u)}, \mathbf{I(u+h)}] &= Prob\{I(\mathbf{u})=1, I(\mathbf{u}+\mathbf{h})=1\} - p^2 \equiv C_I(h) \end{aligned} \tag{3.7}$$

The non-centered covariance of **I(u)** and **I(u+h)**, i.e., E[**I**(u)·**I**(u+h)], can then be rewritten as

$$\begin{aligned} E[\mathbf{I(u)} \cdot \mathbf{I(u+h)}] &= Prob\{I(\mathbf{u})=1, I(\mathbf{u}+\mathbf{h})=1\} \\ &= Prob\{I(\mathbf{u})=1 \mid I(\mathbf{u}+\mathbf{h})=1\} \cdot Prob\{I(\mathbf{u}+\mathbf{h})=1\} \\ &= Prob\{I(\mathbf{u})=1 \mid I(\mathbf{u}+\mathbf{h})=1\} \cdot p \end{aligned} \tag{3.8}$$

where Prob{I(**u**)=1| I(**u+h**)=1} is the conditional probability of the indicator random variable at **u** given that the indicator random variable at **u+h** is 1. Equation (3.8) is the basis for indicator simulation. A suitable regression algorithm, e.g., kriging, can be used to estimate and update the conditional probability of Eq(3.8). Information for the update processes is provided by the available data collected within the neighborhood of the node being estimated.

III.2.1.1(b) Kriging

Kriging is a linear regression algorithm which estimates an SRF at a particular position from the information collected in its neighborhood. It is also called the "best linear unbiased estimator" (BLUE) (Isaaks and Srivastava, 1989). The term "best" is used

because its variance of error is minimized; it is "linear" because its estimates are weighted linear combinations of existing measurements; it is "unbiased" because its mean residual (mean error) is zero. Derivations in this section follow the line given in most textbooks of geostatistics, such as Isaaks and Srivastava (1989).

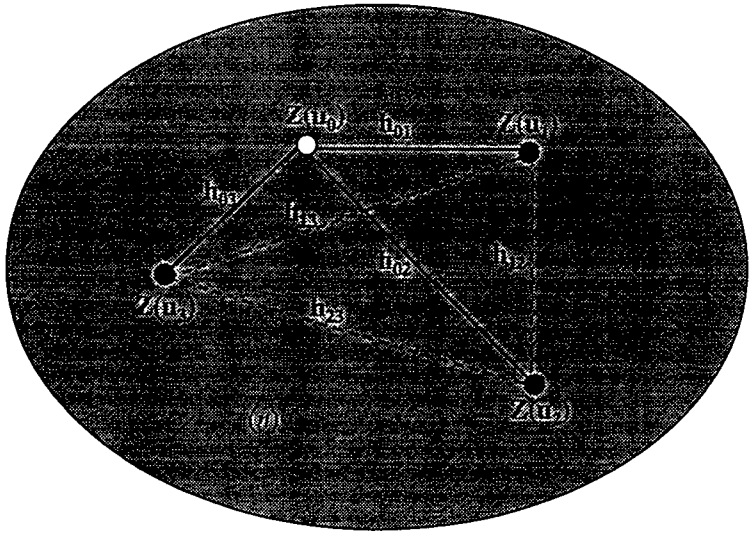


Figure III.1 Sketch of a kriging system with three reference nodes.

For example, in Figure III.1, we want to estimate the variable (**Z**) at an unsampled location, $\mathbf{u}_0$. In kriging, this estimate is written as a weighted linear combination of the measurements from $\mathbf{u}_1$ to $\mathbf{u}_3$, i.e.,

$$\mathbf{Z}_K^*(\mathbf{u}_0) = \beta_0 + \sum_{i=1}^{3} \beta_i(\mathbf{u})\mathbf{Z}(\mathbf{u}_i) \qquad (3.9)$$

where $\mathbf{Z}_K^*(\mathbf{u}_0)$ is the estimate of $\mathbf{Z}$ at $\mathbf{u}_0$, β_0 is a correction term reflecting the measurement bias, β_i are the weights, and $\mathbf{Z}(\mathbf{u}_i)$ are the measurements at location $\mathbf{u}_i$. The subscript K refers to different weighting methods, either simple kriging (*SK*) or ordinary

kriging (*OK*). The difference between simple kriging and ordinary kriging will be the sum of the weights, see the end of this section. Generally, there will be an arbitrary number of measurements such that the upper limit of the summation in Eq(3.9) can be replaced by an arbitrary integer n. Then, a more general equation is

$$\mathbf{Z}_K^*(\mathbf{u}_0) = \beta_0 + \sum_{i=1}^{n} \beta_i(\mathbf{u})\mathbf{Z}(\mathbf{u}_i) \tag{3.10}$$

which simply means that the estimation of $\mathbf{Z}$ at location $\mathbf{u}_0$ is a weighted linear combination of n measurements from $\mathbf{u}_1$, $\mathbf{u}_2$, ..., to $\mathbf{u}_n$, plus an arbitrary constant β_0. The estimation error (Y) of Eq(3.10) is defined as

$$\mathrm{Y} = \mathbf{Z}(\mathbf{u}_0) - \mathbf{Z}_K^*(\mathbf{u}_0) = \mathbf{Z}(\mathbf{u}_0) - \beta_0 - \sum_{i=1}^{n} \beta_i(\mathbf{u})\mathbf{Z}(\mathbf{u}_i) \tag{3.11}$$

where $\mathbf{Z}(\mathbf{u}_0)$ is the true value of $\mathbf{Z}(\mathbf{u})$ at $\mathbf{u}_0$. To ensure that kriging is an unbiased estimator, the expectation of Y must be zero. From Eq(3.11) it yields

$$\mathrm{E}[\mathrm{Y}] = \mathrm{E}\left[\mathbf{Z}(\mathbf{u}_0) - \mathbf{Z}_K^*(\mathbf{u}_0)\right] = m(\mathbf{u}_0) - \beta_0 - \sum_{i=1}^{n} \beta_i(\mathbf{u})m(\mathbf{u}_i) = 0 \tag{3.12}$$

where $m(\mathbf{u}_i) = \mathrm{E}\{\mathbf{Z}(\mathbf{u})\}$ is the location dependent mean values of $\mathbf{Z}$ at $\mathbf{u}_i$. Therefore, the constant β_0 in Eq(3.10) is

$$\beta_0 = m(\mathbf{u}_0) - \sum_{i=1}^{n} \beta_i(\mathbf{u})m(\mathbf{u}_i) \tag{3.13}$$

Substituting Eq(3.13) into Eq(3.10) yields

$$\mathbf{Z}_K^*(\mathbf{u}_0) = m(\mathbf{u}_0) + \sum_{i=1}^{n} \beta_i(\mathbf{u})\left[\mathbf{Z}(\mathbf{u}_i) - m(\mathbf{u}_i)\right] \tag{3.14}$$

which is the estimation equation used in kriging. The estimation error, Y, can then be rewritten as

$$Y = \mathbf{Z}(\mathbf{u}_0) - \mathbf{Z}_K^*(\mathbf{u}_0) = \mathbf{Z}(\mathbf{u}_0) - m(\mathbf{u}_0) - \sum_{i=1}^{n} \beta_i(\mathbf{u})\left[\mathbf{Z}(\mathbf{u}_i) - m(\mathbf{u}_i)\right] = \sum_{i=0}^{n} a_i(\mathbf{u})\left[\mathbf{Z}(\mathbf{u}_i) - m(\mathbf{u}_i)\right] \tag{3.15}$$

where $a_0(\mathbf{u}) = 1$, and $a_i(\mathbf{u}) = -\beta_i$ for $i = 1, \ldots, n$. Then, the variance of Y can be derived as

$$\mathrm{Var}[Y] = \sum_{i=0}^{n}\sum_{j=0}^{n} a_i(\mathbf{u}) a_j(\mathbf{u}) C_{ij}(\mathbf{u}_i, \mathbf{u}_j) \tag{3.16}$$

where $C_{ij}(\mathbf{u}_i, \mathbf{u}_j)$ is the covariance of random variables $\mathbf{Z}(\mathbf{u}_i)$ and $\mathbf{Z}(\mathbf{u}_j)$. To minimize Var[Y], the following system of equations has to be satisfied

$$\begin{aligned} \frac{\partial \mathrm{Var}[Y]}{\partial a_i} &= 2\left[\sum_{j=0}^{n} a_j(\mathbf{u}) C_{ij}(\mathbf{u}_i, \mathbf{u}_j)\right] = 2\left\{a_0(\mathbf{u}_0) C_{i0}(\mathbf{u}_i, \mathbf{u}_0) + \sum_{j=1}^{n} a_j(\mathbf{u}) C_{ij}(\mathbf{u}_i, \mathbf{u}_j)\right\} \\ &= 2\left\{C_{i0}(\mathbf{u}_i, \mathbf{u}_0) - \sum_{j=1}^{n} \beta_j(\mathbf{u}) C_{ij}(\mathbf{u}_i, \mathbf{u}_j)\right\} = 0 \end{aligned} \tag{3.17}$$

The minimum of Var(Y) occurs when

$$\sum_{j=1}^{n} \beta_j(\mathbf{u}) C_{ij}(\mathbf{u}_i, \mathbf{u}_j) = C_{i0}(\mathbf{u}_i, \mathbf{u}_0), i = 1, \cdots, n \tag{3.18}$$

which is called the normal system of equations. For a stationary SRF, the means, $m(\mathbf{u}_i)$, can be written as a constant m. Stationarity is usually the basic assumption of simple

kriging. Therefore, the estimation function, the normal system of equations, and the estimation variance of simple kriging are written as

$$\mathbf{Z}^*_{SK}(\mathbf{u}_0) = m + \sum_{i=1}^{n} \beta_i^{SK}(\mathbf{u})[\mathbf{Z}(\mathbf{u}_i) - m] = \sum_{i=1}^{n} \beta_i^{SK}(\mathbf{u})\mathbf{Z}(\mathbf{u}_i) + \left[1 - \sum_{i=1}^{n} \beta_i^{SK}(\mathbf{u})\right] m \quad (3.19)$$

$$\sum_{j=1}^{n} \beta_j^{SK}(\mathbf{u}) C_{ij}(\mathbf{u}_i - \mathbf{u}_j) = C_{i0}(\mathbf{u}_i - \mathbf{u}_0) \ , \ i = 1, \cdots, n \quad (3.20)$$

$$\begin{aligned} s^2_{SK} &= \mathrm{E}\left[\left(\mathbf{Z}(\mathbf{u}_0) - \mathbf{Z}^*_{SK}(\mathbf{u}_0)\right)^2\right] = \mathrm{E}\left[\left(\sum_{i=0}^{n} a_i \left(\mathbf{Z}_i(\mathbf{u}_i) - m\right)\right)^2\right] \\ &= \sum_{i=0}^{n}\sum_{j=0}^{n} a_i a_j C_{ij}(\mathbf{u}_i - \mathbf{u}_j) = C_{00} - \sum_{i=0}^{n} \beta_i^{SK} C_{i0}(\mathbf{u}_i - \mathbf{u}_0) \end{aligned} \quad (3.21)$$

where $a_0 = 1$, and $a_i = -\beta_i^{SK}$ for $i = 1, \ldots, n$.

The difference in the bracket of the last term in Eq(3.19) is zero if the sum of kriging weights is one. This is one of the requirements for an ordinary kriging system. Thus, the estimation $\mathbf{Z}^*_{SK}(\mathbf{u}_0)$ can be simplified as a linear combination of the n measurements, without the need of knowing the constant mean value m. This constraint can be solved by introducing a Lagrangian multiplier $\Pi(\mathbf{u})$, i.e.,

$$\mathbf{Z}^*_{OK}(\mathbf{u}_0) = \sum_{i=1}^{n} \beta_i^{OK}(\mathbf{u})\mathbf{Z}(\mathbf{u}_i) \quad (3.22)$$

$$\begin{cases} \displaystyle\sum_{j=1}^{n} \beta_j^{OK}(\mathbf{u}) C_{ij}(\mathbf{u}_i - \mathbf{u}_j) - \Pi(\mathbf{u}) = C_{i0}(\mathbf{u}_i - \mathbf{u}_0) \\ \displaystyle\sum_{j=1}^{n} \beta_j^{OK}(\mathbf{u}) = 1 \quad , i = 1, \cdots, n \end{cases} \quad (3.23)$$

$$s_{OK}^2 = \mathrm{E}\left[\left(\mathbf{Z}(\mathbf{u}_0) - \mathbf{Z}_{OK}^*(\mathbf{u}_0)\right)^2\right] = \mathrm{E}\left[\left(\sum_{i=0}^{n} a_i\left(Z_i(\mathbf{u}_i) - m\right)\right)^2\right]$$
$$= \sum_{i=0}^{n}\sum_{j=0}^{n} a_i a_j C_{ij}(\mathbf{u}_i - \mathbf{u}_j) = C_{00} - \sum_{i=0}^{n} \beta_i^{OK} C_{i0}(\mathbf{u}_i - \mathbf{u}_0) + \Pi(\mathbf{u}) \qquad (3.24)$$

where $a_0 = 1$, and $a_i = -\beta_i^{OK}$ for $i = 1, \ldots, n$.

III.2.1.1(c) Indicator kriging

Considering the binary process of asperity contacts versus void space in a fracture plane, the appropriate indicator random variable can be defined as Eq(3.2). To simulate such binary process in space, it is equivalent to asking the following question: what is the conditional probability that the indicator variable at location **u** is 1 given that the indicator variable at location **u+h** is also 1. This is exactly the conditional probability given by Eq(3.8). Thus, indicator kriging is aimed at providing an unbiased estimate of the conditional probability, but not at estimating the indicator variable itself at location **u** (Deutsch and Journel, 1992). For convenience, the conditional probability estimated by indicator kriging is written as $[i(\mathbf{u})]^*$. This conditional probability is equivalent to its conditional expectation because

$$[i(\mathbf{u})]^* \equiv \mathrm{Prob}\{I(\mathbf{u}) = 1 \,|\, I(\mathbf{u}+\mathbf{h}) = 1\} \equiv \mathrm{Prob}\{I(\mathrm{u}) = 1 \,|\, (n)\}^* = \mathrm{E}\{\mathbf{I}(\mathbf{u}) \,|\, (n)\}^* \qquad (3.25)$$

where (n) represents the neighborhood of location **u.** The size of neighborhood grows as more data, either from measurements or recent estimations, become available. The value of $[i(\mathbf{u})]^*$ can be estimated either by simple kriging or ordinary kriging. From the

properties of indicator function, i.e., Eq(3.7), it is known that the mean of an indicator function is its probability of occurrence. Since this probability is assigned prior to indicator simulation, it is appealing to use simple kriging instead of ordinary kriging. Therefore, $[i(\mathbf{u})]^*$ can be estimated by

$$\begin{aligned}[i(\mathbf{u})]^*_{SK} &= \text{Prob}\{I(\mathbf{u})=1 \mid (n)\}^*_{SK} \\ &= \sum_{i=1}^{n} \beta_i^{SK}(\mathbf{u}) I(\mathbf{u}_i) + \left[1 - \sum_{i=1}^{n} \beta_i^{SK}(\mathbf{u})\right] \mathrm{E}\{\mathbf{I}(\mathbf{u})\} \\ &= \sum_{i=1}^{n} \beta_i^{SK}(\mathbf{u}) I(\mathbf{u}_i) + \left[1 - \sum_{i=1}^{n} \beta_i^{SK}(\mathbf{u})\right] \mathrm{p}\end{aligned} \tag{3.26}$$

The normal equations are

$$\sum_{j=1}^{n} \beta_j^{SK}(\mathbf{u}) \mathrm{C_I}(\mathbf{u}_i - \mathbf{u}_j) = \mathrm{C_I}(\mathbf{u}_i - \mathbf{u}_0) \ , \ i = 1, \cdots, n \tag{3.27}$$

where $I(\mathbf{u}_i)$ are realizations of indicator variable in the neighborhood (n), and $\mathrm{C_I}(\mathbf{h}) = \mathrm{Cov}\{\mathbf{I}(\mathbf{u}), \mathbf{I}(\mathbf{u}+\mathbf{h})\}$ is the indicator covariance. If $\mathbf{I}(\mathbf{u})$ is stationary, $\mathrm{C_I}(\mathbf{h})$ is equivalent to $\mathrm{C_I}(\mathrm{h})$.

Indicator simulation starts from a random location, searches the neighborhood of that location to find enough conditioning points for performing kriging, and then updates the conditional probability. This updated conditional probability is compared with a randomly drawn probability (p) to determine the value of the indicator variable at that location. If the random probability is smaller than or equal to p, the indicator variable is set to 1; otherwise it is set to 0. Subsequently, another random path is taken and the above procedures are repeated. Note that the neighborhood (n) for subsequent updates consists of the original data and the previously simulated indicator values. Thus, even if the

indicator simulation starts initially from a null field (no conditioning data at all), it becomes a conditional simulation as long as the neighborhood contains information that was previously simulated. Defining the size of neighborhood for the indicator simulation is equivalent to specifying the search method. An efficient way of searching is not to search all the nodes on the grid but to search a limited number of nodes that are close enough to the node being estimated. The "closeness" is evaluated by the variogram distance (or lag, see section III.2.2), i.e., a node is close to the estimation node if their relative distance is smaller than the variogram distance (Deutsch and Journel, 1992).

III.2.1.2 Spatially random asperity contacts

Boolean simulation is a process that distributes geometric objects in space according to a desired probability law (Deutsch and Journel, 1992). The Boolean algorithm in GSLIB randomly generates two-dimensional ellipses or circles with specified radii, orientations and aspect ratios.

Boolean simulation starts from a random point in space which is the centroid of a geometric object that is going to be formed. The geometric object, either isotropic (a circle) or anisotropic (an ellipse), is constructed by adding "mass" around the centroid until this object satisfies the randomly selected radius, orientation, and anisotropy ratio. Subsequently, another random centroid is chosen and the above procedure is repeated until the specified total fraction of asperity contacts is reached.

In order to generate asperity contacts and simulate the gradual change of aperture away from asperity contacts towards larger aperture between asperities, the original log-

normal sample is shifted to the left by a constant ζ_c, i.e., $\zeta' = \max(\zeta - \zeta_c, 0)$, such that additional asperity contacts are produced. Figure III.2 shows that if ζ_c is 0.63, additional 15% asperity contacts are produced. These additional asperity contacts are un-conditioned data, which may be free to move while being perturbed in a SA system.

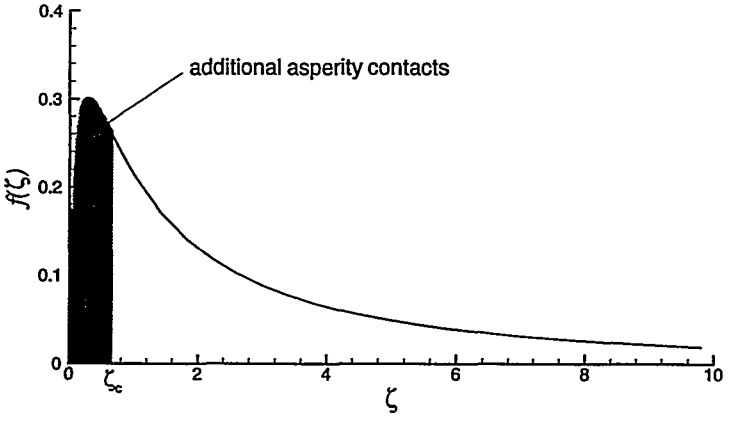


Figure III.2 A log-normal distribution with mean ($m_{\ln\zeta}$) = 1.00 and standard deviation ($s_{\ln\zeta}$) = 1.50. The permeability cutoff, ζ_c, in this plot is 0.63 such that additional 15% asperity contacts (the shaded area) are produced.

III.2.2 Objective function

An objective function, or energy function, is used to transform the SA system into an optimization model. It is a measure of the difference of some spatial features between the desired distribution and the realization. In this study, the objective function is defined as the normalized squared difference of the semi-variogram between the realization and an expected distribution, i.e.,

$$O = \frac{1}{O_{initial}} \left[\sum_{r=1}^{n_{lag}} \frac{\left[\gamma'(\mathbf{h}_r) - \gamma_{expected}(\mathbf{h}_r)\right]^2}{\gamma^2_{expected}(\mathbf{h}_r)} \right] \tag{3.28}$$

where $\gamma'(\mathbf{h}_r)$ is the semi-variogram at separation distance $\mathbf{h}_r$ of the permeability modifier field (ζ), i.e.,

$$\gamma(\mathbf{h}_r) = \frac{1}{2N(h_r)} \sum_{i=1}^{N(h_r)} [Z(u_i) - Z(u_i + h_r)]^2 \equiv \mathrm{E}\left\{[\mathbf{Z(u)} - \mathbf{Z(u+h}_r)]^2\right\} \qquad (3.29)$$

$\gamma_{expected}(\mathbf{h}_r)$ is the expected semi-variogram of ζ, $O_{initial}$ is the objective function of the initial field, and n_{lag} is the total number of lags for $\gamma(\mathbf{h}_r)$. The lag $\mathbf{h}_r$ has to be defined in such a way that the same data pair will not be calculated twice. This definition is illustrated in Figure III.3. The squared difference in Eq(3.28), i.e., $[\gamma'(\mathbf{h}_r) - \gamma_{expected}(\mathbf{h}_r)]^2$, is normalized by $\gamma^2_{expected}(\mathbf{h}_r)$ to give more weight to small values of $\gamma_{expected}(\mathbf{h}_r)$. Note that the term within the outer bracket in Eq(3.28) is further weighted by a factor $1/O_{initial}$. This is for mathematical convenience such that the objection function (O) always starts from 1.

The semi-variogram in Eq(3.28) of the numerical system can be calculated using the following equation

$$\gamma'(\mathbf{h}_r) = \frac{1}{2} \frac{\sum_{r=1}^{N(\mathbf{h}_r)} [\zeta(\mathbf{u}_i) - \zeta(\mathbf{u}_j)]^2}{\sum_{i,j}^{n_{xyz}} \#\text{ of } \left(|\mathbf{u}_i - \mathbf{u}_j| = \mathbf{h}_r\right)} = \frac{1}{2} \frac{\Lambda(\mathbf{h}_r)}{N(\mathbf{h}_r)} \qquad (3.30)$$

where $N(\mathbf{h}_r)$ are the total number of ζ pairs at lag $\mathbf{h}_r$, and $\Lambda(\mathbf{h}_r)$ is the sum of squared differences of $N(\mathbf{h}_r)$ pairs of ζ's. Recall that, in Chapter I, the spatial correlation of the void space as well as asperity contacts can be approximated by an exponential function.

Therefore, it is assumed that the ζ field has an exponential correlation function. Therefore, $\gamma_{expected}(\mathbf{h}_r)$ in Eq(3.28) can be written as

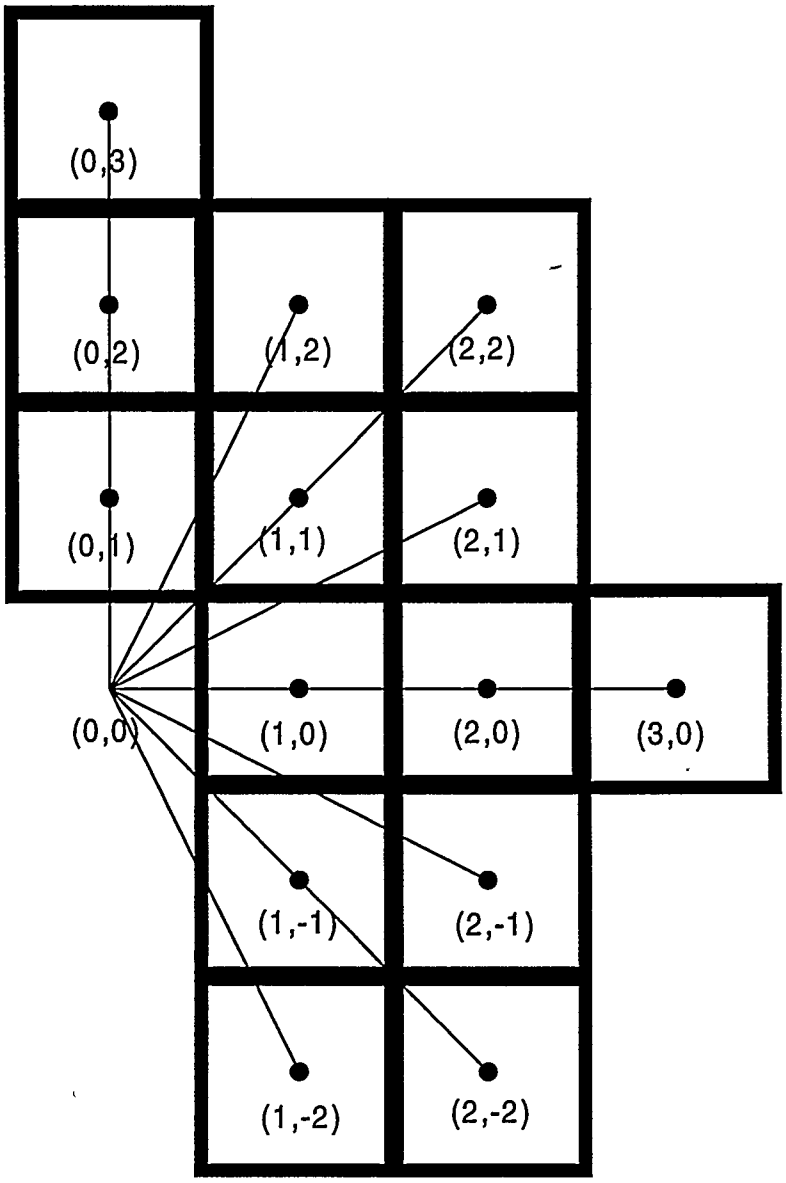
$$\gamma_{\text{expected}}(\mathbf{h}_r) = c + s_\zeta^2\left\{1-\exp\left(-\frac{h_r}{\lambda}\right)\right\} = c + s_\zeta^2\left\{1-\exp\left(-\frac{3h_r}{\xi}\right)\right\} \qquad (3.31)$$

where c, the nugget, is assumed to be zero in this study, s_ζ^2 is the variance of the ζ field, λ is the integral scale (correlation length) in the principal direction, ξ is the range parameter, and h_r is the magnitude of the separation distance $\mathbf{h}_r$. Definitions of λ and ξ can be found in Appendix A. The physical meaning of range is that, at this separation distance, the value of semi-variogram is 95% of s_ζ^2. Or, equivalently, the value of the correlation function is 5% of s_ζ^2. Thus, the random field is practically un-correlated as long as the separation distance is greater than the range (see Appendix A for details). In addition, the range is three times the correlation length for exponential models. The nugget effect is caused by small-scale variability and/or sampling error (Isaaks and Srivastava, 1989), which is explained in Appendix B.

Equation (3.31) is an isotropic semi-variogram. For anisotropic semi-variograms, Eq(3.31) can be modified as

$$\gamma_{\text{expected}}(\mathbf{h}_r) = \gamma_{\text{expected}}(h_1, h_2) = 0 + s_\zeta^2\left\{1-\exp\left(-\sqrt{\left(\frac{h_x}{\lambda_1}\right)^2+\left(\frac{h_z}{\lambda_2}\right)^2}\right)\right\} \qquad (3.32)$$

where the subscripts 1 and 2 denote the longitudinal and transverse axes, and h_1, h_2 as well



$$\begin{cases} \dfrac{\Delta h_{r,x}}{\Delta x} \geq 0 \text{ always} \\ \dfrac{\Delta h_{r,z}}{\Delta z} \neq 0 \text{ when } \dfrac{\Delta h_{r,x}}{\Delta x} > 0 \\ \dfrac{\Delta h_{r,z}}{\Delta z} > 0 \text{ when } \dfrac{\Delta h_{rx}}{\Delta x} = 0 \end{cases}$$

$$\frac{\Delta h_{r,x}}{\Delta x} = \text{horizontal offset}$$

$$\frac{\Delta h_{r,z}}{\Delta z} = \text{vertical offset}$$

Figure III.3 Definition of lag offsets. Lag offsets are assigned such that semi-variogram for a given pair will not be calculated twice. Fourteen lags are illustrated in this plot.

as λ_1, λ_2 are the separation distance and correlation lengths on the longitudinal and transverse axes, respectively. Anisotropy such as in Eq(3.32) is called the "geometric anisotropy" (Isaaks and Srivastava, 1989).

Calculating $\gamma'(\mathbf{h}_r)$ in Eq(3.28) may be a time-consuming task if the grid size is large. To reduce the computational effort, Deutsch and Cockerham (1994) proposed an efficient method of updating $\gamma'(\mathbf{h}_r)$. Since only one random pair is perturbed at a time, $\gamma'(\mathbf{h}_r)$ needs not to be recalculated at each perturbation but can be updated based on previous information. This is illustrated in Figure III.4. Consider the random pair, $\zeta(\mathbf{u}_1)$ and $\zeta(\mathbf{u}_2)$. For a particular lag $\mathbf{h}_r$, the neighboring data points contributing to updating $\gamma'(\mathbf{h}_r)$ are shown as solid circles; whereas data points contributing nothing to updating $\gamma'(\mathbf{h}_r)$ are marked with hollow circles. Therefore, $\gamma'(\mathbf{h}_r)$ can be updated by the following equation

$$\gamma'(\mathbf{h}_r) = \frac{1}{2N(\mathbf{h}_l)} \left[\Lambda(\mathbf{h}_l) + \sum_{i=1}^{2} \left[-(\zeta(\mathbf{u}_1) - \zeta(\mathbf{u}_{1i}))^2 + (\zeta(\mathbf{u}_2) - \zeta(\mathbf{u}_{1i}))^2 \right] + \sum_{i=1}^{2} \left[-(\zeta(\mathbf{u}_2) - \zeta(\mathbf{u}_{2i}))^2 + (\zeta(\mathbf{u}_1) - \zeta(\mathbf{u}_{2i}))^2 \right] \right] \quad (3.33)$$

where $\Lambda(\mathbf{h}_r)$ is the sum of squared difference of ζ pairs from the previous perturbation.

III.2.3 Perturbation mechanism

Starting from the initial field, SA selects a random pair of data points before each perturbation. Each data point in this pair has to be un-conditioned data. The system is then perturbed by comparing the system energy before and after swapping the locations of

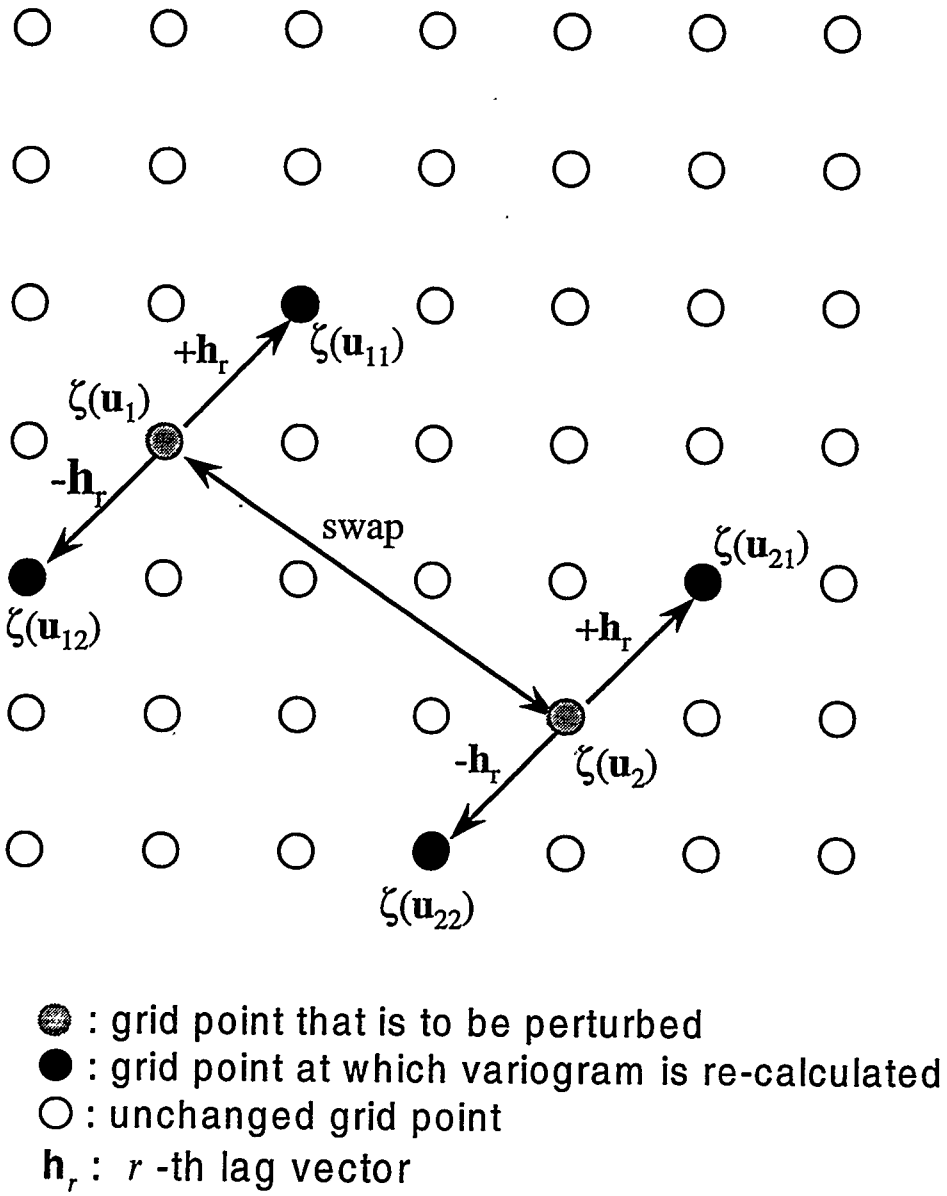


Figure III.4 Update of simulated semi-variogram in a small grid where ($\mathbf{u}_1$, $\mathbf{u}_2$) is the location of the random pair, and $\mathbf{u}_{i1}$ and $\mathbf{u}_{i2}$ are separated from $\mathbf{u}_i$ by lag vectors $+\mathbf{h}_r$ and $-\mathbf{h}_r$, respectively.

the two data points. If the system energy decreases after a perturbation, the two data points will exchange their locations. Otherwise, the system remains unchanged. This process is repeated until the system reaches the state with the minimum energy (O_{min}); or stopped when the number of perturbations is beyond an upper limit. Other perturbation mechanisms may also be used. For example, one possible mechanism is to randomly select a data point and replace its value by a new one drawn from a specific probability population (Datta-Gupta, *et al.*, 1995).

The critical drawback of the above perturbation mechanisms is that the minimum energy O_{min} at convergence may be a local minimum but not global. This is because such perturbation mechanisms always favor the paths with decreasing energy, and the paths with increasing energy are unconditionally rejected. To correct this shortcoming, Metropolis *et al.* (1953) proposed an algorithm such that an unfavorable perturbation can also be accepted with a certain probability. By conditionally accepting an unfavorable perturbation, the system is able to jump out of a local minimum. Then, the optimal system energy at convergence can be close to the global minimum (Press *et al.*, 1986). The perturbation mechanism of unconditionally accepting a favored perturbation but conditionally accepting an unfavorable perturbation has been referred to as the "Metropolis algorithm".

From the theories of thermodynamics and statistical physics, the probability of changing the system energy from E_1 to E_2 can be described by the Boltzmann distribution (Metropolis *et al.*, 1953), i.e.,

$$p' = \exp\left[\frac{-(E_2 - E_1)}{k_B T}\right] = \exp\left[\frac{-\Delta E}{k_B T}\right] \tag{3.34}$$

where k_B is the Boltzmann constant, and T is the temperature. The probability in Eq (3.34) is modified in SA to represent the probability of accepting an unfavorable perturbation. By letting $k_B = 1$, $\Delta E \approx \Delta O$ (change of objective function before and after a perturbation), and T = temperature parameter in SA, the acceptance probability in SA is approximated as

$$p_{accept} \approx \exp\left[\frac{-\Delta O}{T}\right] \tag{3.35}$$

III.2.3.1 Modified Metropolis algorithm

As mentioned earlier in Section III.1, a realistic representation of natural fractures must model the gradual change from zero aperture at asperity contacts toward larger apertures between asperities. However, preliminary tests of the Metropolis algorithm showed that it may not be adequate to achieve that goal because it is not "sensitive enough" to simulating "simply connected" (in the topological sense) asperity contacts. Thus, a modified Metropolis algorithm was developed as a part of this study.

The concept of "neighborhood" was introduced in order to modify the Metropolis algorithm. The neighborhood of an asperity contact is defined as an un-conditioned grid block that the distance from the center of this grid block to the center of the asperity contact is smaller than or equal to l grid block units. For example, the 28 gray blocks shown in Fig. III.5 are the neighborhood of the asperity contact located at (0,0) with $l = 3$.

The purpose of introducing neighborhood to asperity contacts is to treat the regions near and far from asperity contacts separately, and to emphasize particular features of the region near asperity contacts.

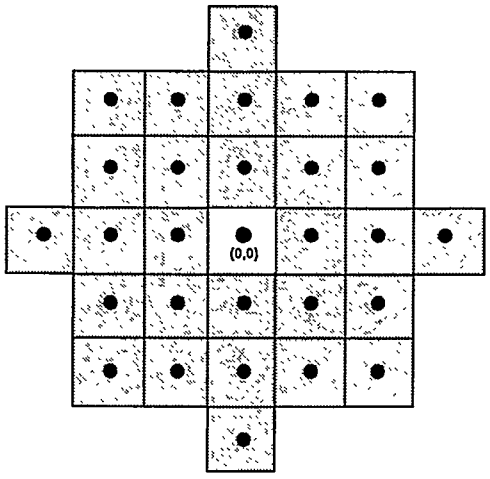


Figure III.5 Schematic definition of the neighborhood illustrated for the asperity contact at (0,0) with $l = 3$.

The Metropolis algorithm is then modified by taking into account the relative locations of the grid blocks in a random pair. If the objective function decreases after a perturbation, this random pair is accepted unconditionally. However, the locations of the random pairs become important if the objective function increases after a perturbation. If both of the grid blocks are in some neighborhoods, or none of them in any neighborhood, this pair is still evaluated probabilistically by Eq(3.35); otherwise, this pair is accepted only if the grid block located in a neighborhood has a larger value of ζ than the grid block that is not in any neighborhood. In other words, it is favored to introduce grid blocks with small values of ζ into neighborhoods of asperity contacts. Figure III.6 illustrates the idea of the modified Metropolis algorithm. Also, see Appendix C for the source code of the

modified Metropolis algorithm. Note that the modified Metropolis algorithm reduces to the Metropolis algorithm if the size of neighborhood (l) is zero.

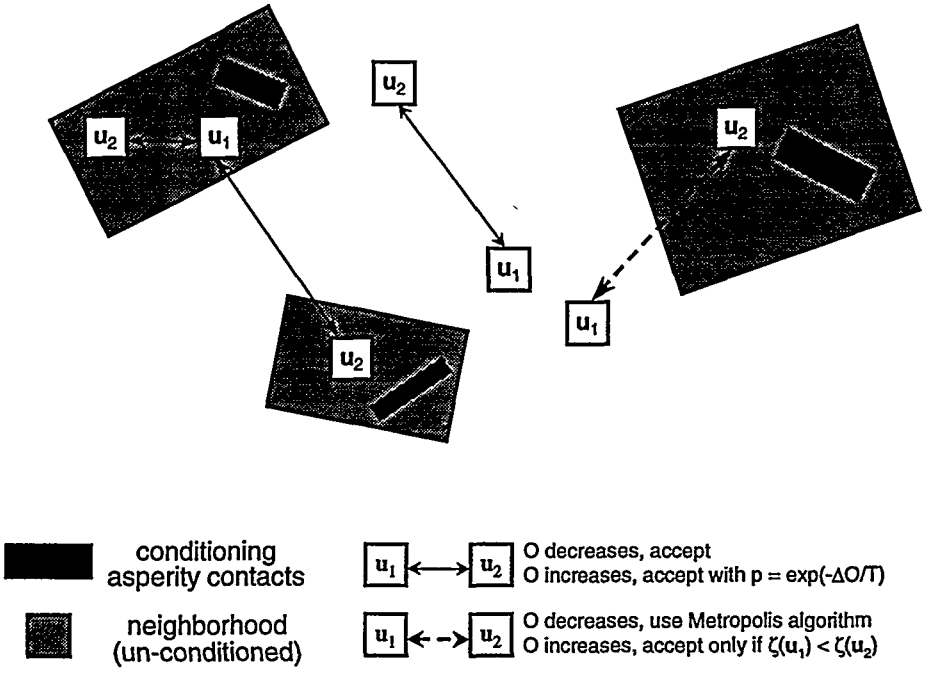


Figure III.6 Concepts of the modified Metropolis algorithm.

III.2.4 Annealing schedule

While annealing a material, the temperature in the thermodynamic system is lowered gradually until the system reaches the state with the minimum thermal energy. To numerically simulate the thermodynamic processes of annealing, it is then necessary to define, in the numerical system, a controlling parameter which acts like the real temperature in the thermodynamic system. In the SA algorithm, the controlling parameter is also called "temperature". Thus, the annealing schedule is the specification of the timing and magnitude of the temperature reduction in the numerical system.

Since the temperature in a real thermodynamic system is lowered continuously, it reflects that all particles in the system experience the same temperature as they cool down. In SA systems, however, only one random pair is chosen at each perturbation. To model the simultaneous temperature reduction analogous to a thermodynamic system, the temperature parameter in SA systems has to be lowered piecewise but not continuously. Numerically, this can be done by lowering the temperature when either one of the following two conditions is satisfied: (1) the number of favored perturbations exceeds the upper limit, κ_{accept}; or (2) the total number of perturbations (either favored or rejected perturbation) after the previous perturbation at which the temperature is lowered exceeds the maximum allowable value, κ_{max}. If one of the above two conditions is satisfied, the temperature will be lowered by a factor τ ($\tau < 1$).

A suitable annealing schedule should be chosen such that T_0 is as large as possible, and τ is as small as possible. However, such annealing schedule is at the expense of large amount of computations, especially when the grid is finely discretized. Therefore, a compromise between good annealing results and a reasonable computational effort is necessary. Based on our experience, the following annealing schedule is satisfactory, i.e., $T_0 = 1.0$, $\tau = 0.9$, $\kappa_{max} = 50\ n_{xyz}$, $\kappa_{accept} = 5\ n_{xyz}$, $\eta = 3$, $O_{min} = 10^{-7}$, which are initial temperature, temperature reduction factor, maximum number of allowable perturbations between two consecutive reductions of temperature, maximum number of accepted perturbations, the maximum allowable ratio of the number of perturbations to κ_{max} when the objective function continues to increase after each perturbation, and the

minimum objective function, respectively. The number n_{xyz} is the total number of grid blocks.

III.3 Effects of spatial discretization on characterization of random fields

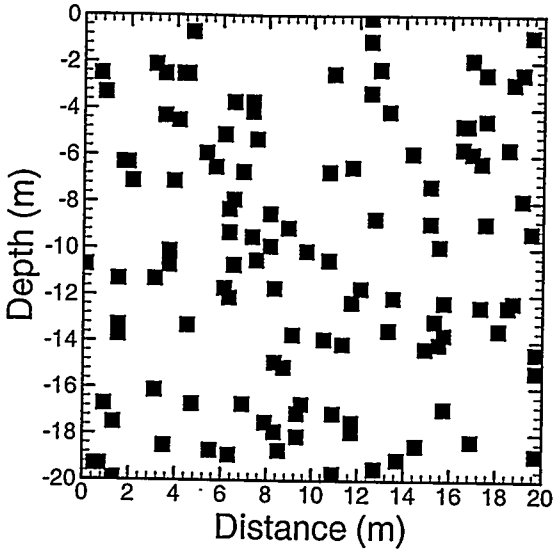
In general, spatial discretization of the numerical grid should be as fine as possible to capture the detailed spatial variability of the random field. However, fine discretization may often make burdensome the computation loads of numerical characterization and flow simulation. Moreover, preferential flow is commonly observed in unsaturated flow in fractured rocks. This suggests that some areas in the fracture will not even be contacted by the aqueous phase due to flow bypassing. Therefore, from the computational point of view, using a fine discretization may not be as cost-effective for characterization purposes as for flow modelings. Thus, a "reasonable" spatial discretization should be adopted. The value of this spatial discretization should be chosen such that basic elements of spatial variability of permeability are preserved, and the resulting flow simulation is physically meaningful as well as representative of field conditions. In this study, the size of the flow domain is 20 m × 20 m × 1 cm. Considering computational capacity and efficiency, a suitable spatial discretization was then chosen as Δ = 0.2 m, i.e., totally 10,000 grid blocks. For comparison purposes in this section, a finer discretization of Δ = 0.1 m is also considered.

One of the factors controlling the dependence of the accuracy of the generated random field on spatial discretization (Δ) is the correlation length (λ) of the random field. A dimensionless ratio of spatial discretization to correlation length, Δ/λ, is commonly

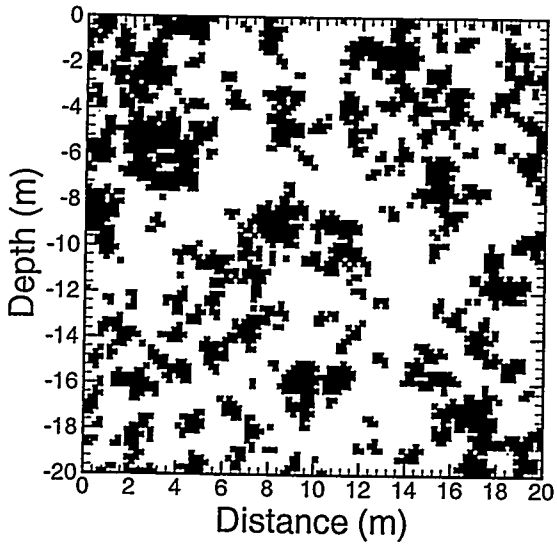
used to analyze the relationship between characterization accuracy and spatial discretization. A theoretical study by Li and Der Kiureghian (1993) suggested that a simulated random field has negligible error with respect to its true random field if this ratio is smaller than 0.5, i.e., $\Delta/\lambda \leq 0.5$. Detailed analyses of the relationship between the spatial discretization and the accuracy of characterization is not pursued in this study. Instead, a sensitivity analysis is performed in this section to examine the impact of spatial discretization on characterization accuracy. Another set of sensitivity analysis on the effect of spatial discretization on seepage patterns is latter investigated in Chapter V.

Two realizations of conditioning asperity contacts that are spatially random and spatially correlated are shown in Figures III.7(a) and III.7(b), respectively. Radius (r_0) and correlation length of asperity contacts (λ_0) are both 0.4m. Based on these asperity contacts, Figure III.8 shows corresponding realizations of permeability fields with different spatial discretizations and correlation lengths (λ_k) of permeability. Note that the same asperity contacts, whether spatially random or spatially correlated, are used both in a coarse grid (Δ = 0.2 m) and a fine grid (Δ = 0.1 m). Two correlation lengths of permeability are considered, λ_k = 0.4 m and 0.8 m. Thus, the ratio Δ/λ_k is 0.25 or 0.5, both satisfying the requirement of $\Delta/\lambda_k \leq 0.5$. Modified Metropolis algorithm was used for annealing each of the permeability fields in Figure III.8. In addition, the correlation was given by an isotropic exponential semi-variogram, Eq(3.31).

It is expected that certain spatial features of the permeability field may become apparent as the ratio Δ/λ_k is decreased. For permeability fields with spatially random

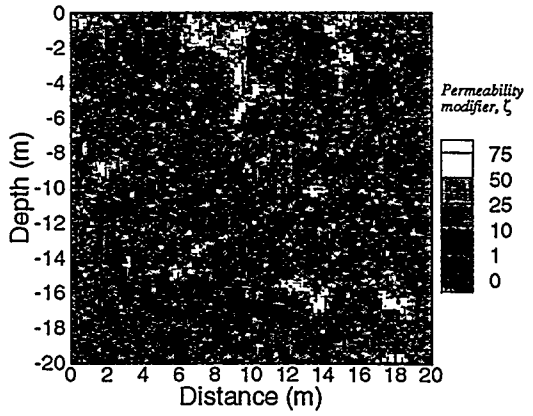


(a) isotropic, spatially random asperity contacts with radius $r_0 = 0.4$m

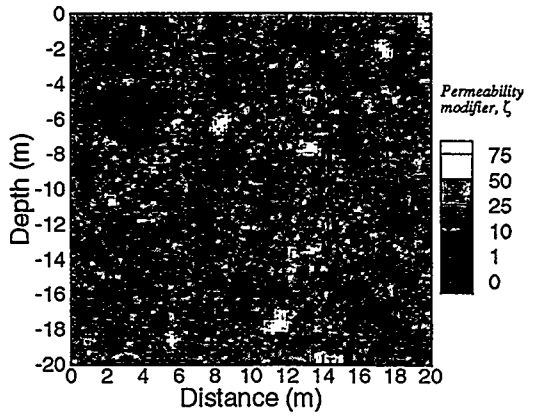


(b) isotropic, spatially correlated asperity contacts with $\lambda_0 = 0.4$m

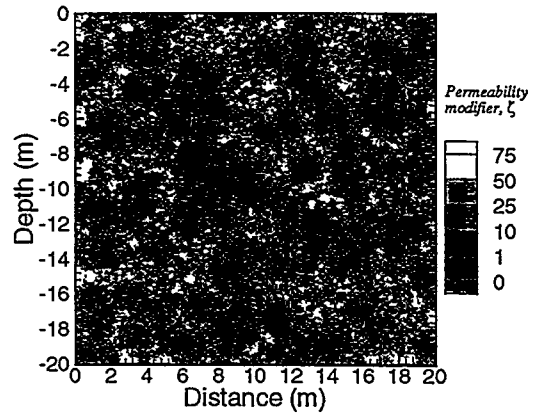
Figure III.7 Two realizations of spatially random and spatially correlated asperity contacts in a grid discretized with $\Delta = 0.2$m.



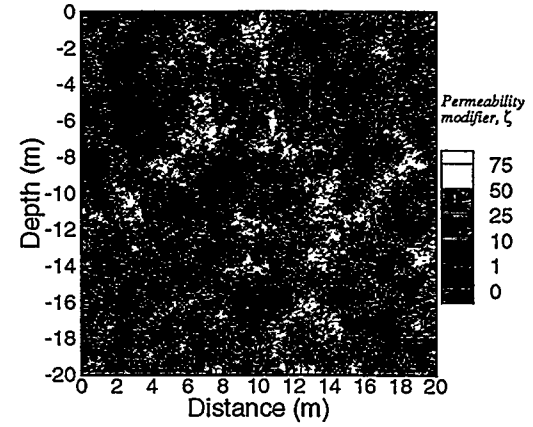
(a) $\Delta = 0.2$m, $r_0 = 0.4$m, $\lambda_k = 0.4$m ($\Delta/\lambda_k = 1/2$)



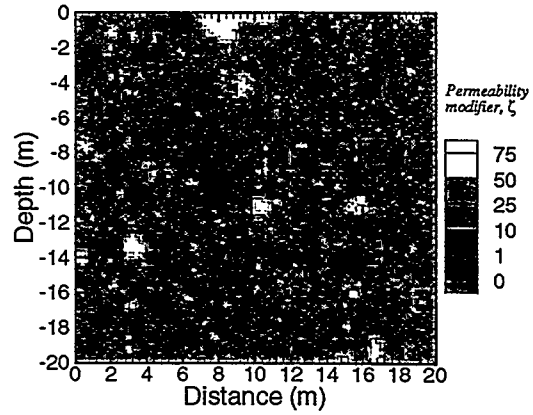
(d) $\Delta = 0.2$m, $\lambda_0 = 0.4$m, $\lambda_k = 0.4$m ($\Delta/\lambda_k = 1/2$)



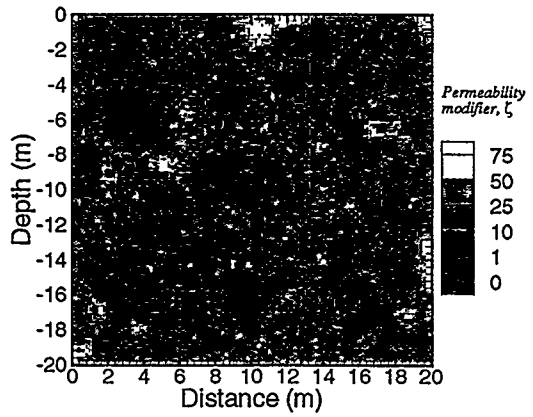
(b) $\Delta = 0.1$m, $r_0 = 0.4$m, $\lambda_k = 0.4$m ($\Delta/\lambda_k = 1/4$)



(e) $\Delta = 0.1$m, $\lambda_0 = 0.4$m, $\lambda_k = 0.4$m ($\Delta/\lambda_k = 1/4$)



(c) $\Delta = 0.2$m, $r_0 = 0.4$m, $\lambda_k = 0.8$m ($\Delta/\lambda_k = 1/4$)



(f) $\Delta = 0.2$m, $\lambda_0 = 0.4$m, $\lambda_k = 0.8$m ($\Delta/\lambda_k = 1/4$)

Figure III.8 Permeability fields used for the sensitivity analysis. Each permeability field is annealed with the modified Metropolis algorithm in which realizations (a) to (c) are conditioned on spatially random asperity contacts (Figure III.7(a)), and realizations (d) to (f) on spatially correlated asperity contacts (Figure III.7(b)). The correlation structure for each realization is an isotropic exponential semi-variogram with nugget = 0, sill = 190 (for realizations (a) to (c)) or 110 (for realizations (d) to (f)), and correlation lengths (λ_k) = 0.4m or 0.8m.

asperity contacts, Figures III.8(a) and III.8(b) show that the clustering effect around asperity contacts becomes more obvious as the ratio Δ/λ_k is decreased from 0.5 to 0.25. Accordingly, flow channeling is expected to be more prominent in Figure III.8(b) than in Figure III.8(a). Comparisons between Figures III.8(d) and III.8(e), however, demonstrate that the intrinsic variability of a random field with spatially correlated asperity contacts is nearly preserved as long as the ratio Δ/λ_k is smaller than 0.5, though finer resolution of tortuosity is observed in a fine grid, Figure III.8(e).

As stated earlier, it is not practical to further refine the grid. Thus, the other way of looking at the impact of the ratio Δ/λ_k on characterization (and/or flow simulation) is to fix the spatial discretization at Δ = 0.2 m but increase the correlation length of permeability. Figures III.8(c) and III.8(f) demonstrate this change by increasing λ_k from 0.4 m to 0.8 m. For spatially correlated asperity contacts, Figures III.8(d) and III.8(f) show that heterogeneities in these two realizations with different ratios of Δ/λ_k are qualitatively the same. However, permeability heterogeneity for random fields with spatially random asperity contacts varies with the ratio Δ/λ_k, see Figures III.8(a) and III.8(c).

Combining the above observations concludes that fracture heterogeneity is virtually insensitive to the ratio Δ/λ_k for permeability fields with spatially correlated asperity contacts, as long as the ratio $\Delta/\lambda_K \leq 0.5$. For permeability fields with spatially random asperity contacts, permeability heterogeneity is sensitive to the ratio Δ/λ_k. As far as the accuracy of characterization is considered, a finer grid may be needed while considering permeability field with spatially random asperity contacts.

III.4 Examples of simulated random fields

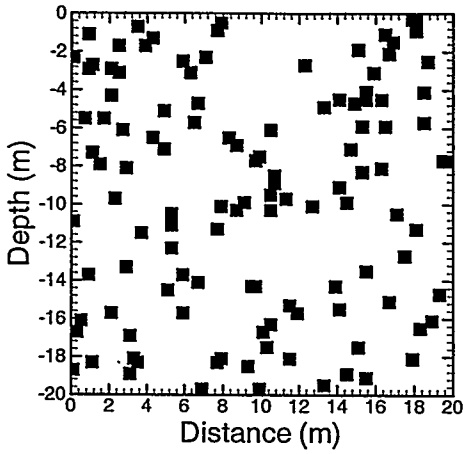
To demonstrate the effectiveness of the modified Metropolis algorithm, Figure III.9 shows four realizations of spatially random as well as spatially correlated conditioning asperity contacts, which are either isotropic or anisotropic. Corresponding initial fields are shown in Figure III.10. Note that the un-conditioned asperity contacts in Figure III.10 were drawn from a log-normal distribution with $m_{\ln\zeta} = 1.0$ and $s_{\ln\zeta} = 1.5$.

Figures III.11(c) and (d) show two permeability fields annealed with the Metropolis and the modified Metropolis algorithms, respectively. The spatial structure is an anisotropic exponential semi-variogram with nugget = 0.0, sill = 120, and correlation lengths (λ_k) = 1.6 m and 0.2 m in the major and minor axes, respectively. Compared with permeability fields annealed with the Metropolis algorithm, permeability fields annealed with the modified Metropolis algorithm have a stronger tendency to draw un-conditioned asperity contacts and/or grid blocks with smaller values of ζ into the neighborhoods of asperity contacts. This tendency is independent of the spatially correlation of asperity contacts. In addition, the tendency may often be obtained at the expense of a larger number of perturbations, compare Figures III.11(a) and III.11(b). Thus, Figure III.11(d) shows a more significant clustering effect around asperity contacts than Figure III.11(c). Moreover, these two realizations can be quantitatively compared by defining the average permeability in the neighborhood as follows

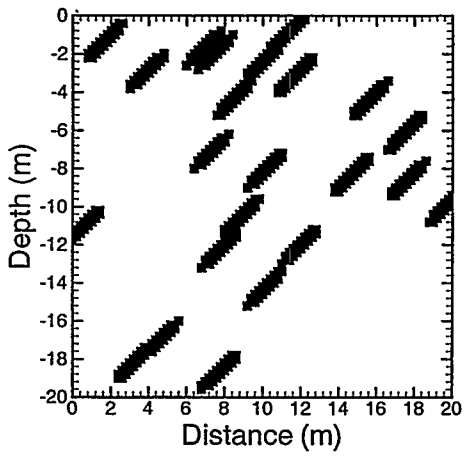
$$k_{\text{avg}} = \frac{1}{\text{Nb}} \sum_{i=1}^{\text{Nb}} k(\mathbf{X}_i) \tag{3.36}$$

where Nb is the total number of grid blocks that are located in the entire neighborhood of all asperity contacts. The average permeability is significantly reduced from 6.7×10^{-9} m^2 in Figures III.11(c) to 5.4×10^{-9} m^2 in Figure III.11(d). Thus, this provides quantitative confirmation that the modified Metropolis algorithm is better suited to our problem than the Metropolis algorithm.

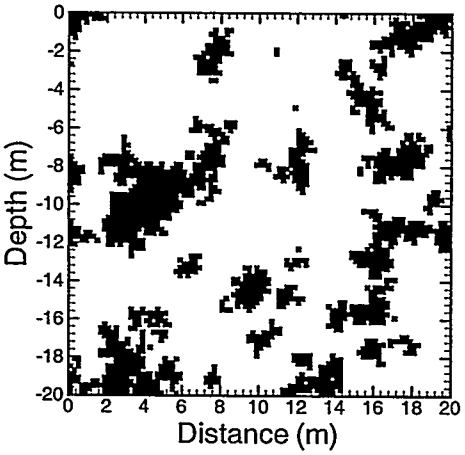
Figures III.11(a) and III.11(b) show the change of objective function with respect to the number of perturbation, corresponding to permeability fields annealed with the Metropolis and modified Metropolis algorithm, respectively. The curve in Figure III.11(a) shows a monotonic decreasing trend. However, the curve in Figure III.11(b) shows significant fluctuation before convergence. Even if the Metropolis algorithm considered the possibility of taking an unfavorable path while perturbing the random field, Figure III.11(a) suggests that it still tends to get trapped in a local minimum. The fluctuating curve in Figure III.11(b) implies that permeability fields annealed with the modified Metropolis algorithm is more likely to reach the global minimum energy. Although realizations in Figure III.11(c) and III.11(d) reveal distinctive clustering effects, the semi-variogram at the end of SA, Figure III.11(e) and III.11(f), both fit the expected correlation structure. Figure III.12 shows permeability fields annealed with the modified Metropolis algorithm, corresponding to initial fields in Figure III.10. Corresponding plots of change of objective function and semi-variogram are shown in Figure III.13. Again these plots show that the modified Metropolis algorithm is able to produce significant clustering effect around conditioning asperity contacts as well as perturb the permeability field to the desired spatial correlation.



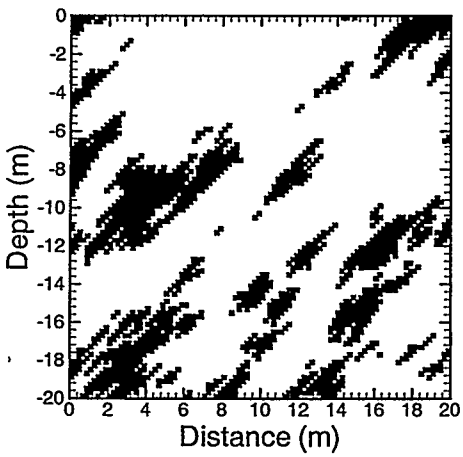
(a) isotropic, spatially random, $r_0 = 0.4$m



(b) anisotropic, spatially random, $r_0 = 1.6$m and 0.4m in NE-SW and NW-SE, respectively

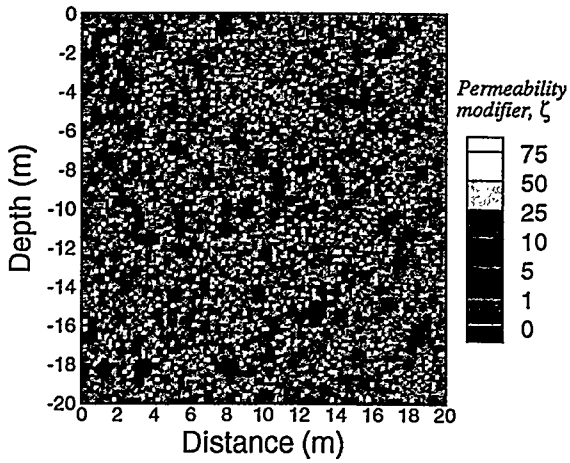


(c) isotropic, spatially correlated, $\lambda_0 = 0.8$m

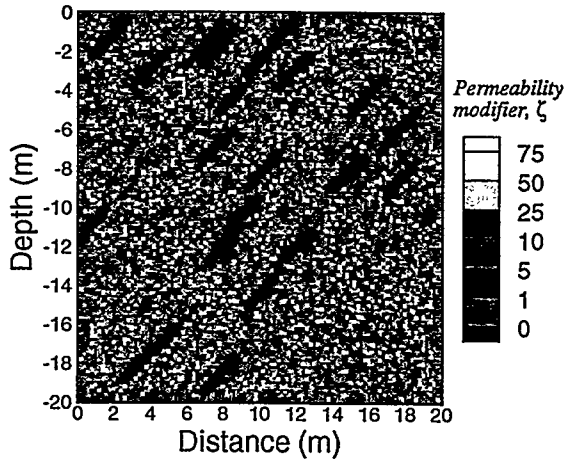


(d) anisotropic, spatially correlated, $\lambda_0 = 1.6$m and 0.4m in NE-SW and NW-SE, respectively

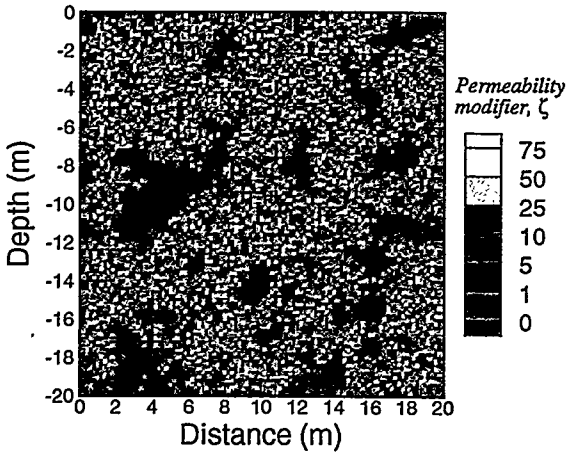
Figure III.9 Examples of spatially random (a and b) and spatially correlated (c and d) asperity contacts. The correlation function of asperity contacts for realizations (c) and (d) are two exponential semi-variograms with nugget = 0, sill = 0.1875, but different correlation scales (λ_0)



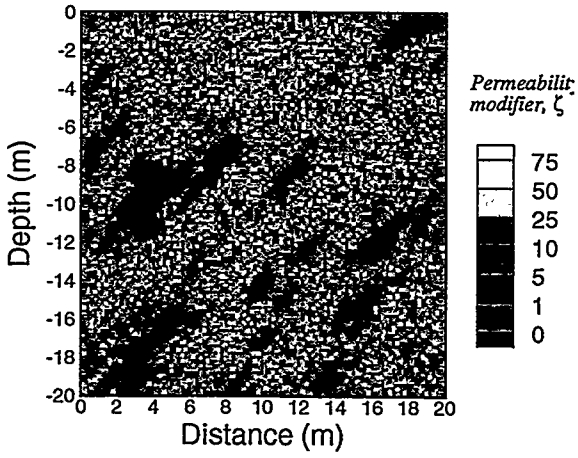
(a) initial permeability field corresponding to conditioning asperity contacts in Figure III.9(a)



(b) initial permeability field corresponding to conditioning asperity contacts in Figure III.9(b)



(c) initial permeability field corresponding to conditioning asperity contacts in Figure III.9(c)



(d) initial permeability field corresponding to conditioning asperity contacts in Figure III.9(d)

Figure III.10 Initial permeability fields corresponding to each of the conditoning asperity contacts shown in Figure III.9. A log-normal distribution with mean and standard deviation of $\ln\zeta$ as 1.0 and 1.5, respectively, was used to generate the un-conditioned data.

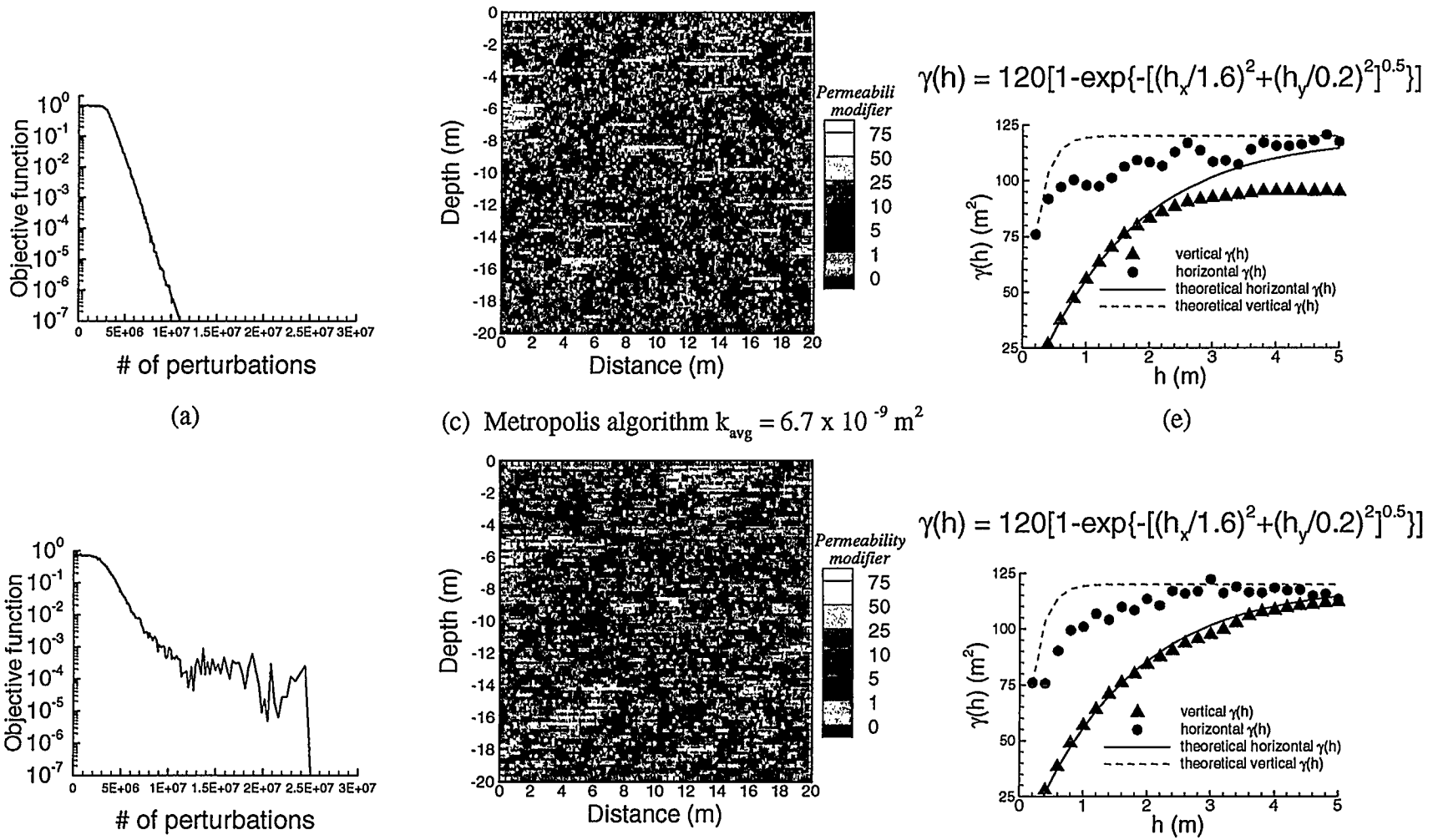


Figure III.11 Permeability fields annealed with the Metropolis and the modified Metropolis algorithms. Each realization was annealed with an anisotropic exponential semi-variogram with nugget = 0, sill = 120, and correlation lengths (λ_k) = 1.6 m and 0.2 m in the major and minor axes, respectively.

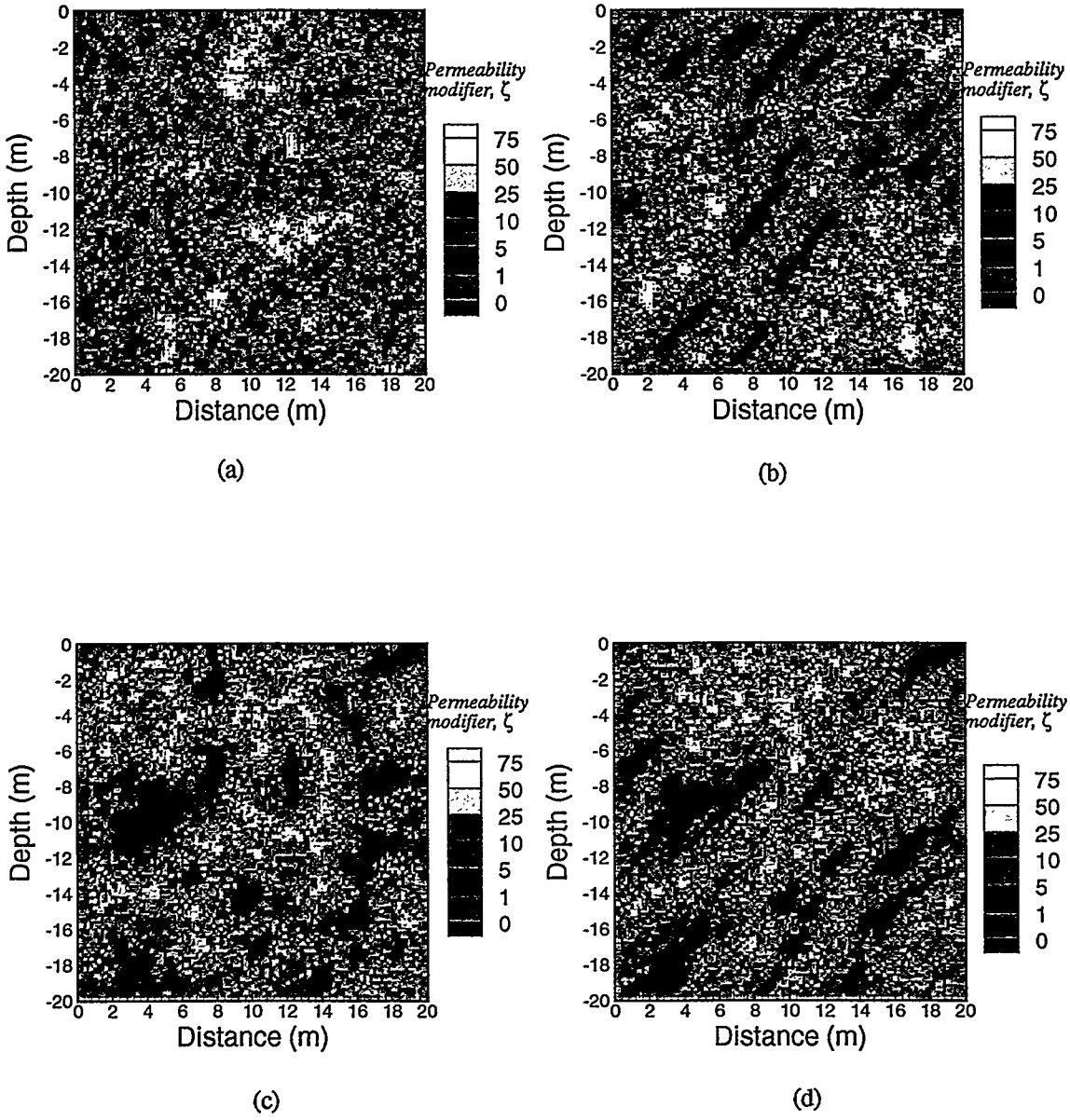
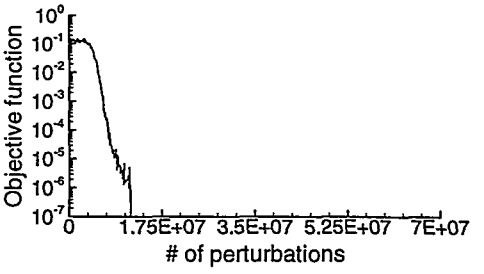
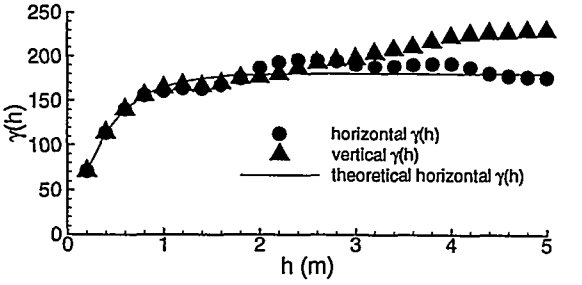


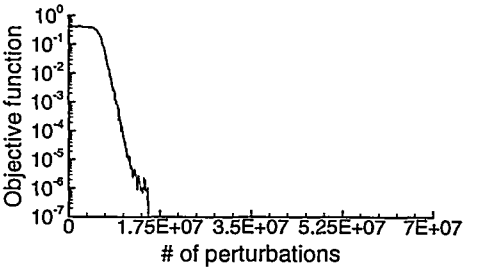
Figure III.12 Annealed permeability fields corresponding to the initial fields shown in Figure III.10 Each permeability field was annealed with the modified Metropolis algorithm. The correlation structure was an isotropic exponential semi-variogram with nugget = 0, correlation length (λ_k) = 0.4m, and sill = 180.0 for realizations a and b, and 120.0 for realizations c and d. Spatial discretization (Δ) = 0.2m.



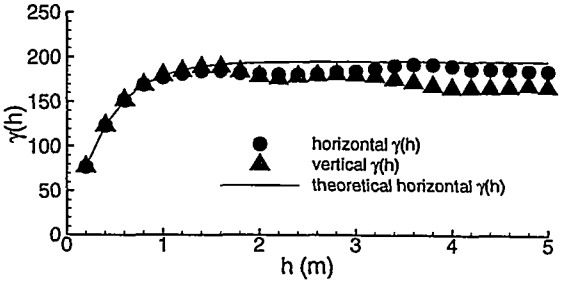
(a-1) permeability field (a)



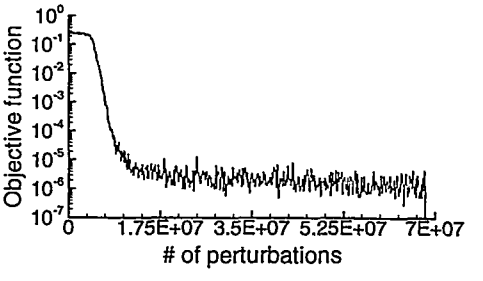
(a-2) permeability field (a) with sill=180.0



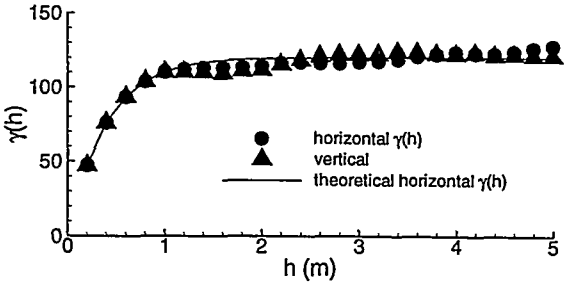
(b-1) permeability field (b)



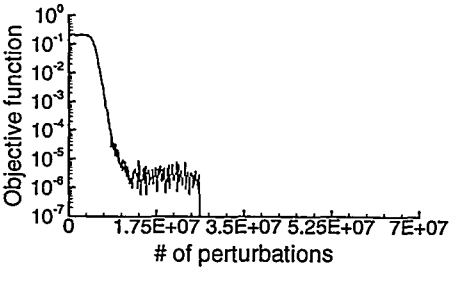
(b-2) permeability field (b) with sill=195.0



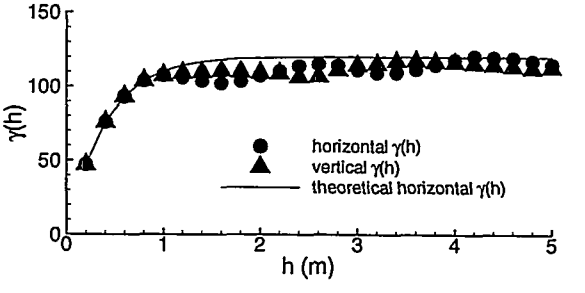
(c-1) permeability field (c)



(c-2) permeability field (c) with sill=120.0



(d-1) permeability field (d)



(d-2) permeability field (d) with sill=120.0

Figure III.13 Change of objective function during annealing, and simulated semi-variograms after annealing for permeability fields corresponding to those shown in Figure III.12

Chapter IV. Flow simulation

IV.1 Numerical simulator

A general-purpose flow simulator, TOUGH2 (Pruess, 1991), is used in this study. TOUGH2 is a numerical simulation program for nonisothermal flow of multicomponent, multiphase fluids in porous and fractured media. The acronym "TOUGH" stands for "Transport Of Unsaturated Groundwater and Heat". TOUGH2 is able to simulate a variety of flow problems by substituting suitable fluid property modules into the modularized architecture. Each flow module specifies the hydrological/thermal properties of fluids under consideration, which is also referred to as "equation-of-state" or EOS module. Thus, TOUGH2 is applicable to a wide range of problems including geothermal reservoir engineering (O'Sullivan, *et al.*, 1998), nuclear waste isolation (Senger, *et al.*, 1998), environmental contamination (Webb, *et al.*, 1998), unsaturated zone hydrology (Doughty, 1998), and mining engineering (Xu *et al.*, 1998)

IV.2 Governing equation

The problem considered in this study is strictly a two-phase (water and air) flow under partially saturated, isothermal conditions in naturally fractured rocks. By making proper assumptions and approximations (see Chapter I.2) this problem reduces to a single-phase flow problem in equivalent 2-D heterogeneous porous media. Furthermore, fluid properties such as density as well as viscosity can be treated as constants under isothermal conditions. Based on these assumptions and approximations, the equation-of

state flow module reduces to that of solving the mass balance equation (the Richards' equation) of the aqueous phase in partially saturated porous media.

In addition to the neglect of gas phase pressure, several assumptions are implied in Richards' equation (Philip, 1969). First, the continuum approach must be valid such that hydrologic parameters can be represented as values that are averaged over a representative elemental volume (REV), a volume that is "large" enough compared to an individual pore but is "small" relative to some macroscale. Second, Darcy's law must hold. That is, inertia effects must be negligible and fluid properties are Newtonian. Third, the flow is isothermal. Once thermal effects become significant, vapor diffusion may turn out to be an important mechanism. For non-isothermal systems, an additional balance equation of heat must be solved along with the liquid phase balance equation. Thus, the Richard's equation can be written in a multi-phase form as follows

$$\frac{\partial}{\partial t}(\phi S_l \rho_l) = \nabla \cdot \left[k \frac{k_{rl}}{\mu_l} \rho_l \nabla (P_l + \rho_l g z) \right] \tag{4.1}$$

where ϕ is porosity, S_l is liquid saturation, ρ_l is liquid density, k is the absolute permeability, k_{rl} is the liquid phase relative permeability, μ_l is liquid viscosity, P_l is the liquid phase pressure, g is gravity, and z is the elevation. Liquid saturation (S_l) is defined as

$$S_l = \frac{V_l}{\phi V_b} = \frac{\theta_l}{\phi} \tag{4.2}$$

in which θ_l is the volumetric moisture content of the liquid phase, i.e.,

$$\theta_l = \frac{V_l}{V_b} \qquad (4.3)$$

where V_l is the volume of the liquid phase and V_b is the bulk volume of the medum. Equation (4.1) is a volume-averaged equation though the scale at which these average values are taken is not explicitly recognized. It is, however, assumed that Eq (4.1) and its implied approximations, such as applicability of relative permeability and capillary pressure, are valid for the equivalent porous media on a scale of 0.1 – 1 m (Pruess, 1998).

Recall that fractured media are similar to porous media in terms of relative permeability and capillary pressure (Persoff and Pruess, 1995). If hysteresis effects are neglected, relative permeability as well as capillary pressure can be expressed in terms of a single-valued function of liquid saturation only. Accordingly, customary formulas of relative permeability and capillary pressure for porous media, such as van Genuchten's equations, can be used in Eq (4.1), which are

$$k_{rl} = \sqrt{S^*}\left\{1-\left(1-\left[S^*\right]^{1/\omega}\right)^{\omega}\right\}^2$$
$$P_{cap} = -\left(\frac{\rho_l g}{\Omega}\right)\left(\left[S^*\right]^{1/\omega}-1\right)^{1-\omega} \qquad (4.4)$$
$$S^* = (S_l - S_{lr})/(1-S_{lr})$$

where S_{lr} is the residual saturation of the liquid phase, and S^* is a scaled saturation such that it is in the range [0, 1]. Parameters used in Eq (4.4) correspond to those for coarse sands, i.e., $\omega = 0.457$, $S_{lr} = 0.15$ for k_{rl} and 0.0 for P_{cap}, and $\Omega = 50\ m^{-1}$ for reference permeability (k_{ref}) at $10^{-9}\ m^2$. These two functions are illustrated in Figure IV.1.

Following the sign convention of P_{cap} in Eq (4.4), the liquid phase pressure in Eq (4.1) can then be written as $P_l = P_{gas} + P_{cap}$, in which P_{gas} is the constant gas phase pressure.

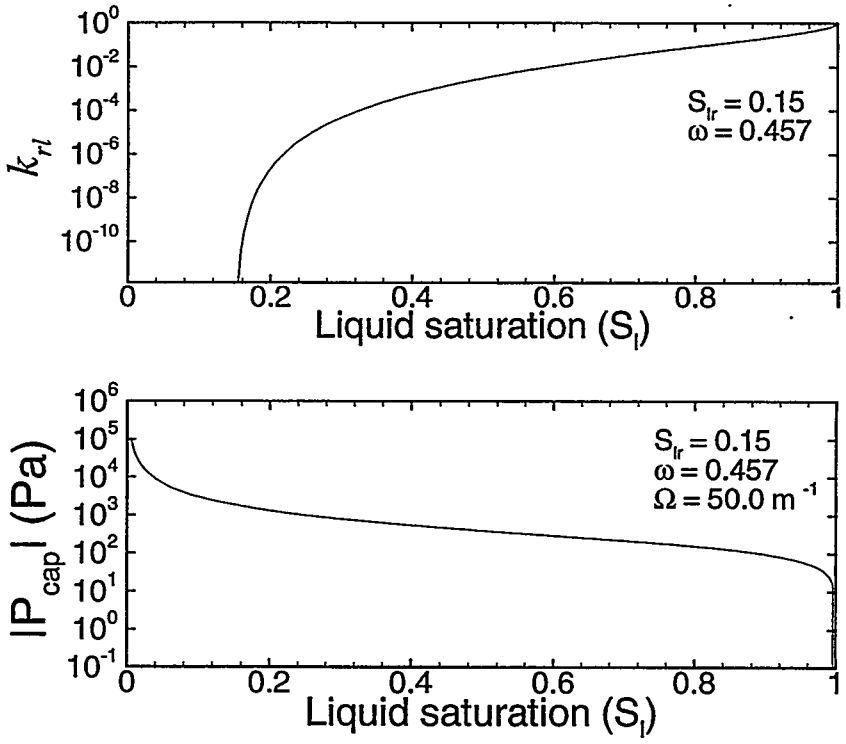


Figure IV.1 Relative permeability and capillary pressure from van Genuchten's formulas, with parameters chosen for coarse sands.

IV.3 Integral form of Richards' equation

Richards' equation is conventionally written as a differential form such as Eq (4.1), in which a divergence operator is included. The shortcoming of using a differential form of balance equations is that the expression of the divergence operator changes with coordinate systems. However, the inherent physical quantity should be invariant to coordinate systems. Therefore, an integral form of Eq (4.1) is preferred. This is the basic idea of the integral finite difference scheme (Narasimhan and Witherspoon, 1976), which is used throughout TOUGH2. Integral finite difference avoids any reference to a global

system of coordinates, and offers the advantage of being applicable to regular or irregular discretization in one, two, or three dimensions (Pruess, 1991).

Discretizing the flow domain into N elements, Eq (4.1) can be rewritten as the following integral form

$$\frac{d}{dt}\int_{V_n} M^{(i)}dV=\int_{\Gamma_n} \mathbf{F}^{(i)}\mathbf{n}d\Gamma+\int_{V_n} q^{(i)}dV \tag{4.5}$$

where V_n is the volume of element n in the discretized flow domain, $M^{(i)}$ is the mass of component i in V_n, $\mathbf{F}^{(i)}$ is the mass flux of component k across the element boundary Γ_n associated with a normal vector $\mathbf{n}$, and $q^{(i)}$ is the mass sink/source term of component i in element n. Each term in Eq (4.5) can be further decomposed as follows. $M^{(i)}$ is the accumulated mass of component i in all phases, i.e.,

$$M^{(i)} = \phi\sum_{\beta=1}^{p} S_\beta X_\beta^{(i)} \rho_\beta \tag{4.6}$$

where p is the total number of phases, S_β is the saturation of phase β, $X_\beta^{(i)}$ is the mass fraction of component k present in phase β, and ρ_β is the fluid density of phase β. Mass flux $\mathbf{F}^{(i)}$ is the sum of fluxes of component i from all phases, i.e.,

$$\begin{aligned} \mathbf{F}^{(i)} &= \sum_{\beta=1}^{p} \mathbf{F}_\beta \\ \mathbf{F}_\beta &= -\frac{kk_{r\beta}}{\mu_\beta}\rho_\beta X_\beta^{(i)}\left(\nabla P_\beta - \rho_\beta \mathbf{g}\right) \end{aligned} \tag{4.7}$$

where $\mathbf{F}_\beta$ is the total mass flux of phase β, μ_β is the viscosity of phase β, k is the intrinsic permeability, $k_{r\beta}$ is the relative permeability of phase β, P_β is the pressure of phase β, and $\mathbf{g}$ is gravity. Note that $\mathbf{F}_\beta$ is the multiphase version of Darcy's law.

Each integral in Eq (4.5) can be approximated by a product of element volume (V_n) and a volume-averaged variable. The volume integral of mass $M^{(i)}$ can be approximated as

$$\int_{V_n} M^{(i)} dV = V_n M_n^{(i)} = V_n \sum_{\beta=1}^{p} \phi_n S_{n,\beta} \rho_{n,\beta} X_{n,\beta}^{(i)} \tag{4.8}$$

where $M_n^{(i)}$, ϕ_n, $S_{n,\beta}$, $\rho_{n,\beta}$, and $X_{n,\beta}^{(i)}$ are volume-averaged values of $M^{(i)}$, ϕ, S_β, ρ_β, and $X_\beta^{(i)}$ within element n, respectively. Of course, Eq (4.8) is valid only if the discretization is fine enough so that $M_n^{(i)}$, ϕ_n, $S_{n,\beta}$, $\rho_{n,\beta}$, and $X_{n,\beta}^{(i)}$ are uniformly distributed within element n. By the same token, the surface integral of interface mass flux $\mathbf{F}^{(i)}$ can be approximated as

$$\int_{\Gamma_n} \mathbf{F}^{(i)} \mathbf{n} d\Gamma = \sum_m A_{nm} F_{nm}^{(i)} = \sum_m A_{nm} \left(\sum_{\beta=1}^{p} F_{\beta,nm} \right) \tag{4.9}$$

The term $\mathbf{F}_{nm}^{(i)}$ in Eq (4.9) is the averaged interfacial mass flux across the interface A_{nm} between element n and all its contacting elements m. This mass flux vector is illustrated in Figure IV.2 as pointing from element m to element n. For simplicity, only one contacting element is shown in Figure IV.2. In addition, the interfacial mass flux $\mathbf{F}_{nm}^{(i)}$ is actually a summation of mass fluxes from various phases. Thus, $\mathbf{F}_{nm}^{(i)}$ is further

decomposed as the last summation in Eq (4.9). From Eq (4.7), $F_{nm}^{(i)}$ can be further decomposed as

$$F_{nm}^{(i)} = -\sum_{\beta=1}^{p} X_{\beta}^{(i)} k_{nm} \left(\frac{k_{r\beta}}{\mu_{\beta}}\right)_{nm} \rho_{\beta} \left(\frac{P_{\beta,n} - P_{\beta,m}}{L_{nm}} - \rho_{\beta,nm} g_{nm}\right) \tag{4.10}$$

where k_{nm}, $\frac{k_{r\beta}}{\mu_{\beta}}$, and $\rho_{\beta,nm}$ are, respectively, the interfacial permeability, mobility, and density associated with the interface A_{nm}. The last integral in Eq (4.5) is simply approximated as $V_n q_n^{(i)}$ in which $q_n^{(i)}$ is the volume-averaged sink/source.

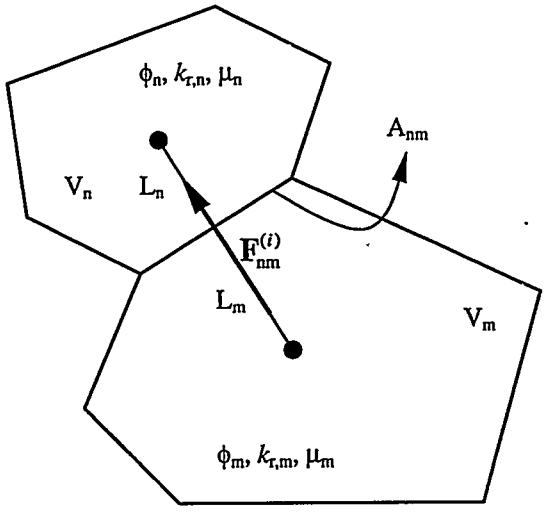


Figure IV.2 Interfacial mass flux $F_{nm}^{(i)}$ across the interface A_{nm} and associated parameters in elements n and m.

Different weighting schemes are used in TOUGH2 for calculating interfacial parameters. Interfacial permeability is harmonically weighted depending on distances L_n and L_m, i.e.,

$$k_{nm} = \begin{cases} \dfrac{k_n \times k_m (L_n + L_m)}{L_m k_n + L_n k_m} \text{ if } L_n \neq 0 \text{ and } L_m \neq 0 \\ k_m \text{ if } L_n = 0 \\ k_n \text{ if } L_m = 0 \end{cases} \tag{4.11}$$

Interfacial density can be upstream weighted, i.e.,

$$\rho_{nm} = WM_n \times \rho_n + WM_m \times \rho_m \tag{4.12}$$

where $WM_n = 1$ and $WM_m = 0$ if the driving force is directed toward n; otherwise, $WM_n = 0$ and $WM_m = 1$. Or it can be uniformly weighted, i.e.,

$$\rho_{nm} = \begin{cases} 0.5 \times \rho_n + 0.5 \times \rho_m \text{ if } L_n \neq 0 \text{ and } L_m \neq 0 \\ \rho_{nm} = \rho_m \text{ if } L_n = 0 \\ \rho_{nm} = \rho_n \text{ if } L_m = 0 \end{cases} \tag{4.13}$$

In order to obtain physically realistic results, interfacial mobility must be upstream weighted (Peaceman, 1977), i.e.,

$$\left(\frac{k_r}{\mu}\right)_{nm} = WM_n \times \frac{k_{rn}}{\mu_n} + WM_m \times \frac{k_{rm}}{\mu_m} \tag{4.14}$$

where WM_n and WM_m are defined above.

In summary, the integral form of Richards' equation can be approximated as

$$\frac{dM_n^{(i)}}{dt} = \frac{1}{V_n} \sum_m A_{nm} F_{nm}^{(i)} + q_n^{(i)} \tag{4.15}$$

Equation (4.15) is solved by the finite difference scheme. In order to obtain numerical stability, time is discretized fully implicitly (Peaceman, 1977). Therefore, Eq (4.15) is recast as

$$M_n^{(i),j+1} - M_n^{(i),j} = \frac{\Delta t}{V_n}\left\{\sum_m A_{nm}F_{nm}^{(i),j+1} + V_n q_n^{(i),j+1}\right\} \qquad (4.16)$$

where j is the index of time stepping. The difference of the two terms to the left and to the right of the equal sign is referred to as :"residual". Therefore, solution of Eq (4.16) occurs when the residuals are zero. Eq (4.16) can then be rearranged as

$$R_n^{(i),j+1}(\Theta) = M_n^{(i),j+1}(\Theta) - M_n^{(i),j}(\Theta) - \frac{\Delta t}{V_n}\left\{\sum_m A_{nm}F_{nm}^{(i),j+1}(\Theta) + V_n q_n^{(i),j+1}\right\} = 0 \qquad (4.17)$$

where R is the residual. For each volume element V_n there are NEQ equations. For a flow system with N grid blocks, Eq (4.17) thus represents a system of NEQ × N coupled, non-linear and algebraic equations. The vector Θ in Eq (4.17) contains NEQ × N independent primary variables which completely define the state of the flow system at time level t^{j+1}. Expanding Eq (4.17) by its Taylor's series to the first order yields

$$R_n^{(i),j+1}(\Theta_{\upsilon+1}) = R_n^{(i),j+1}(\Theta_\upsilon) + \sum_{u=1}^{N\times NEQ}\left.\frac{\partial R_n^{(i),j+1}}{\partial \Theta_u}\right|_\upsilon (\Theta_{u,\upsilon+1} - \Theta_{u,\upsilon}) = 0 \qquad (4.18)$$

where υ is the iteration index. Eq (4.18) can be solved by the iterative Newton-Raphson method. Time step in Eq (4.18) may be automatically adjusted, depending on the condition of convergence during the iteration processes (Pruess, 1991). Usually, Eq

(4.18) is written in the following form

$$-\sum_{u=1}^{N\times NEQ}\left.\frac{\partial R_n^{(i),j+1}}{\partial \Theta_u}\right|_{\upsilon}\left(\Theta_{u,\upsilon+1}-\Theta_{u,\upsilon}\right)=R_n^{(i),j+1}(\Theta_{\upsilon}) \tag{4.19}$$

which is further simplified as

$$\mathbf{J}\Delta\Theta=\mathbf{R} \tag{4.20}$$

where **J** is the Jacobian matrix of the flow system (the partial derivatives of R_n with respect to Θ_i) and **R** is the residual vector consisting of NEW × NEL residuals. Other features and flexibility of TOUGH2 code can be seen in detail in user's guides for TOUGH and TOUGH2 (Pruess, 1987, 1991).

III.4 Verification of TOUGH2 – One-Dimensional Infiltration Model

An example of one-dimensional infiltration into a semi-infinite porous medium was used to verify the TOUGH2 code. The movement of water in an one-dimensional, semi-infinite unsaturated soil column can be described by the following form of the Richards' equation (Philip, 1955)

$$\frac{\partial\theta}{\partial t}=\frac{\partial}{\partial x}\left(D\frac{\partial\theta}{\partial x}\right)-\frac{\partial K}{\partial x} \tag{4.21}$$

where K is hydraulic conductivity, and D is the diffusivity that is defined as

$$D=K\frac{d\,h}{d\theta} \tag{4.22}$$

where h is the hydraulic head, which is defined as the sum of pressure head (P/ρg) and elevation head (z), i.e.,

$$h = \frac{\mathrm{P}}{\rho g} + z \tag{4.23}$$

The first term in (4.21) represents the capillary effect, and the second term gravity effect. Equation (4.21) can be further simplified by neglecting gravity effect. This assumption is valid either at earlier times of infiltration or for horizontal soil tubes. Thus Eq (4.21) can be reduced to

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial x}\left(\mathrm{D}\frac{\partial \theta}{\partial x}\right) \tag{4.24}$$

Assume that the soil column has an initial water content at θ_n at $x > 0$. In addition, water is infiltrated into the soil surface at $x = 0$ at a constant water content θ_0. These conditions can be written as

$$\begin{aligned} \theta &= \theta_n \ , \ t = 0, x > 0 \\ \theta &= \theta_0 \ , \ x = 0, t \geq 0 \end{aligned} \tag{4.25}$$

and are illustrated in Figure IV.3.

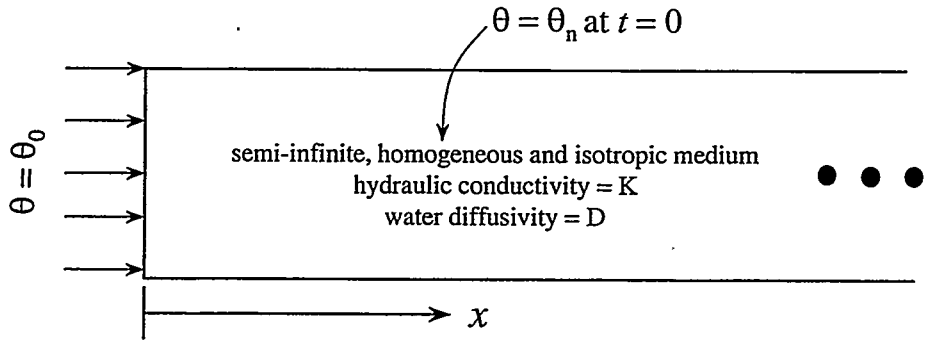


Figure IV.3 Schematic diagram of the one-dimensional infiltration problem.

Based on similarity transformation (or Boltzmann transformation), Philip (1955) developed an iterative analytical solution for Eq (4.24) subject to initial and boundary conditions Eq (4.25). He transformed Eq (4.24) into an ordinary differential equation by introducing a new variable, φ, that is a function of θ only. The number of independent variables in Eq (4.24) is then reduced to one if φ is written as

$$\varphi = \varphi(\theta) = x \cdot t^{-1/2} \tag{4.26}$$

Thus, Eq (4.24) can be recast as

$$-\frac{\varphi}{2}\frac{d\theta}{d\varphi} = \frac{d}{d\varphi}\left(D\frac{d\theta}{d\varphi}\right) \tag{4.27}$$

Multiplying both sides of Eq (4.27) by $\frac{d\varphi}{d\theta}$ gives

$$-\frac{\varphi}{2} = \frac{d}{d\theta}\left(D\frac{d\theta}{d\varphi}\right) \tag{4.28}$$

Initial condition and boundary condition for Eq (4.28) can be deduced from Eq (4.25) as

$$\begin{aligned} &\theta = \theta_0 \ , \ \varphi = 0 \\ &\theta \to \theta_n \ , \ \varphi \to \infty \end{aligned} \tag{4.29}$$

Eq (4.29) implies that

$$\frac{d\varphi}{d\theta} \to 0 \text{ as } \theta \to \theta_n \tag{4.30}$$

Integrating Eq (4.28) with respect to θ then yields

$$\int_{\theta_n}^{\theta} \frac{\varphi}{2} d\theta = -2D \frac{d\theta}{d\varphi} \tag{4.31}$$

which is subject to the condition

$$\theta = \theta_0 \ , \ \varphi = 0 \tag{4.32}$$

Philip's method is to solve Eq (4.31) with the condition specified in Eq (4.32). Figure IV.4 illustrates an example of the relationship between φ and θ. This relationship is obtained by assuming that relative permeability and capillary pressure of soil are described by van Genuchten's formulas, Eq (4.4), with parameters corresponding to coarse sands. Once the function $\varphi(\theta)$ is found, other relationship such as saturation profile or infiltration rate can be derived from that function.

III.4.1 Test problem

Consider a horizontal soil tube with semi-infinite extent, e.g., Figure IV.3. Assume that this soil has a porosity of 0.45 and intrinsic permeability of 1.2×10^{-14} m^2. Initially, water saturation (S_l) in this soil tube is 44%. Then, water is infiltrated into the soil tube at $x = 0$ until it is fully saturated, i.e., $S_l = 1$. Relative permeability (k_r) and capillary pressure (P_{cap}) for this soil are assumed by the following linear functions

$$k_r = \begin{cases} 0 \ , \ S_l < 0.333 \\ \dfrac{S_l - 0.333}{0.667} \ , \ 0.333 \le S_l \le 1 \end{cases} \tag{4.33}$$

$$P_{cap} = \begin{cases} -9.7902\times10^3 \text{ (Pa)} \ , \ S_l \le 0.333 \\ -9.7902\times10^3 \dfrac{1-S_l}{0.667} \text{ (Pa)}, \ 0.333 < S_l < 1 \\ 0 \ , \ S_l \ge 1 \end{cases} \tag{4.34}$$

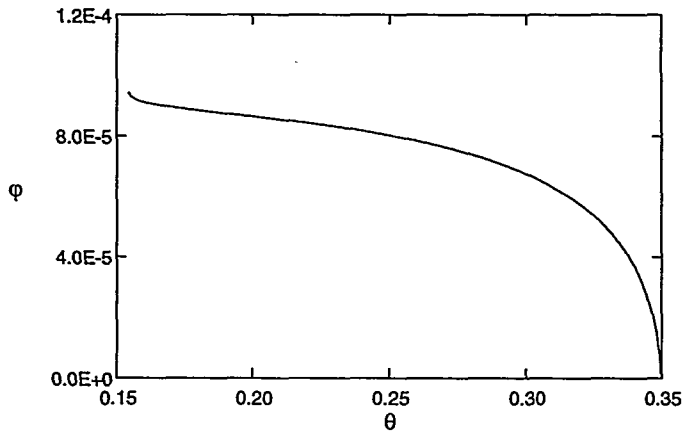


Figure IV.4 Functional form of φ versus θ for a soil whose relative permeability and capillary pressure functions are described by van Genuchten's formulas.

which are illustrated in Figure IV.5. From Eq (4.33), the hydraulic conductivity, K, can be written as the product of k and k_r, where k is the absolute permeability. Then, Eq (4.22) can be rewritten as

$$D = K\frac{dh}{d\theta} = \frac{kk_r}{\mu\phi}\frac{dP_{cap}}{dS} \tag{4.35}$$

where μ is water viscosity (10^{-3} Pa·s). From Eqs(4.33) and (4.34), Eq (4.35) is

$$D\ (m^2/s) = \begin{cases} 5.8683\times10^{-7}(S_l - 0.333)\ ,\ 0.333 < S_l < 1 \\ 0\ ,\ S_l < 0.333 \end{cases} \tag{4.36}$$

The analytical solution, i.e., $\phi(\theta)$, is shown in Figure IV.6. Based on this relationship, the saturation profile at a particular time can be obtained by multiplying $\varphi(\theta)$ by $\sqrt{t}$. For example, Figure IV.7 shows three saturation profiles at t = 864 sec, 5184 sec, and 9504 sec. Solutions obtained by TOUGH2 are marked by symbols, and

analytical solutions by lines. It is clear from Figure IV.7 that analytical solutions obtained by Philip's method are very close to numerical solutions by TOUGH2.

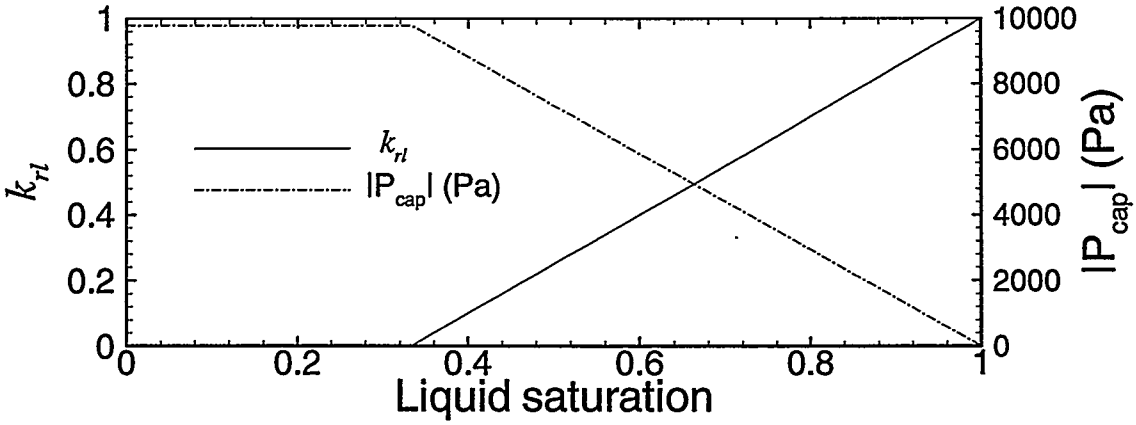


Figure IV.5 Linear relative permeability and absolute capillary pressure for the soil considered in the test problem.

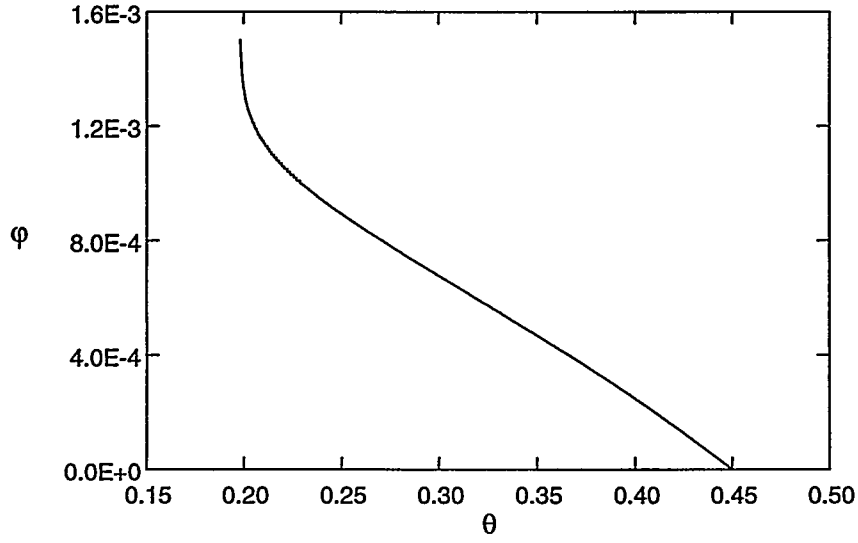


Figure IV.6 Analytical solution obtained by Philip's method for the one-dimensional infiltration problem with θ_n=0.198 and θ_0=0.45.

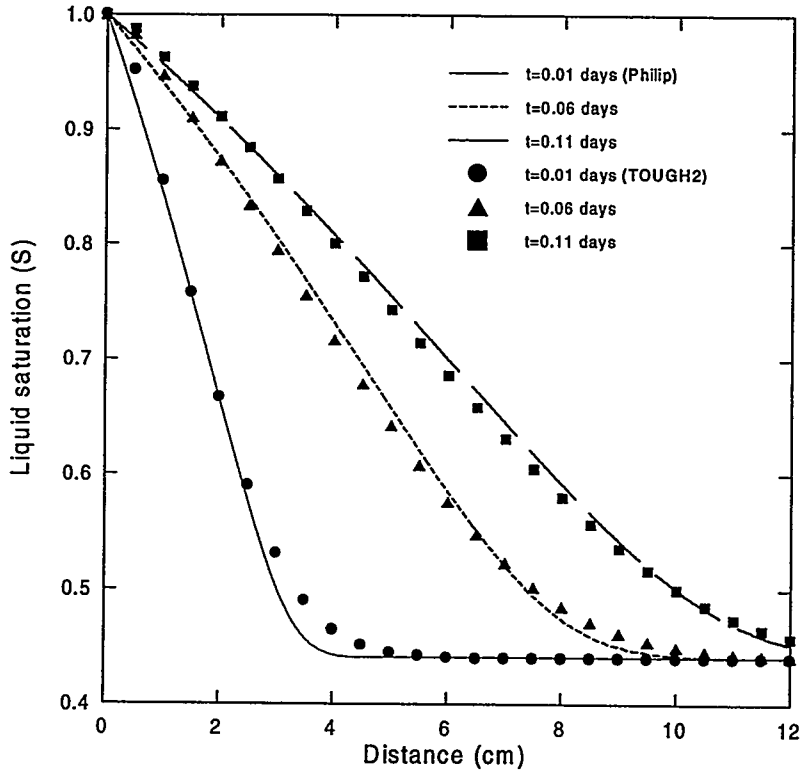


Figure IV.7 Comparison of analytical and numerical solutions for one-dimensional infiltration into a horizontal soil tube.

Chapter V. Seepage simulations

V.1 Setup of numerical simulations and factors affecting seepage in fractures

Fluid flow in a partially saturated fracture is examined in this chapter, considering different levels of normal stress, varying degrees of fracture heterogeneity, and various initial and boundary conditions. Transient as well as steady state flow is considered.

The flow domain is idealized as in Figure V.1 in which the equivalent porous medium lies within a vertically oriented fracture plane of 20 m × 20 m × 1cm. This domain is further discretized into a finite difference grid of 100 × 100 = 10,000 square blocks. Such discretization is fine enough to represent medium heterogeneity, and is manageable with the available computers. Porosity heterogeneity is neglected. Accordingly, a spatially uniform porosity (ϕ) of 0.35 is assumed. Water is injected into the top boundary with a constant rate (10^{-3} kg/s), which is done by introducing an additional single element (20 m × 0.2 m × 1 cm) at the ground surface (z = 0). This element not only receives the water supply but transfers fluid mass with underlying elements. Initial liquid saturation in fractures is assumed at the value of the residual liquid saturation, i.e., S_{lr} = 0.15. Lateral boundaries have no-flow boundary conditions. In addition, a unit-gradient boundary is assumed at the bottom boundary, i.e., the free drainage boundary condition. The above initial and boundary conditions are applied to most simulations if no other conditions are specified.

Effects of the gas phase pressure, matrix permeability, porosity heterogeneity, and hysteresis of relative permeability and capillary pressure are neglected in this study. The

remaining factors affecting unsaturated seepage in fractures, therefore, are the heterogeneity and spatial correlation of permeability, as well as the initial and boundary conditions. For homogeneous porous media, capillary pressure is inversely proportional to a length scale that characterizes the pore structure of the media (Leverett, 1941), i.e.,

$$P_{cap} \propto 1 \Big/ \sqrt{\frac{k}{\phi}} \tag{5.1}$$

where k and ϕ are the permeability and porosity of the media. For heterogeneous media, however, both permeability and porosity may be spatially varying variables. Since the porosity is assumed to be homogeneous, capillary pressure for heterogeneous media is thus scaled as

$$P'_{cap} = \sqrt{k_1 / k_2}\, P_{cap} = \sqrt{1/\zeta}\, P_{cap} \tag{5.2}$$

in which k_1 and k_2 are the permeabilities of the equivalently homogeneous and heterogeneous media, which are k_{ref} and $k_{ref} \times \zeta$, respectively.

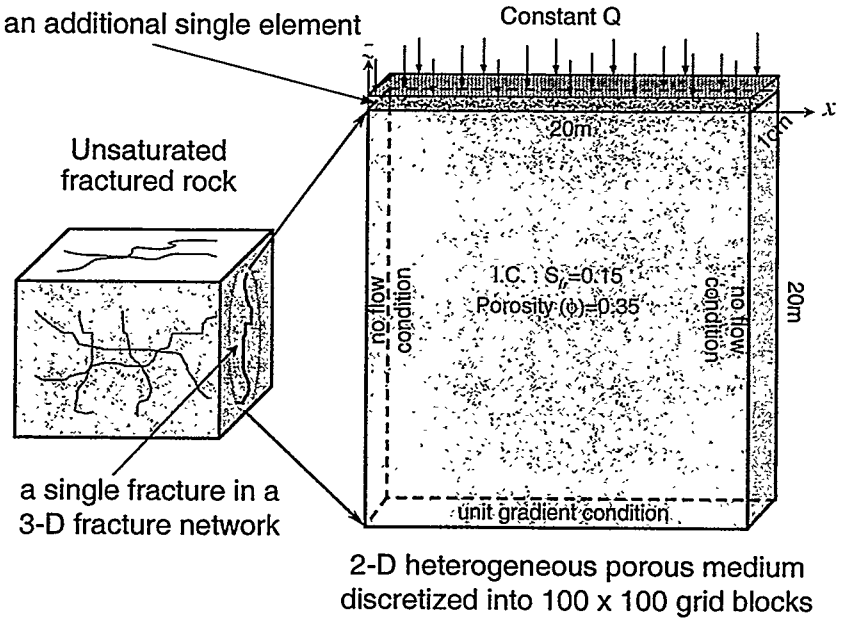


Figure V.1 Idealization of the flow domain.

The effect of permeability on seepage depends not only on the aperture, which is a function of the magnitude of normal stress (see Figure II.2), but also anisotropy of asperity contacts. The latter is conceptually illustrated in Figure V.2, which shows laterally extensive asperity contacts perpendicular, parallel, or oblique to the downward flow direction. Asperity contacts are represented as ellipses in Figure V.2 for simplicity and convenience, though they may not necessarily have regular shapes. Figure V.2(a) shows that asperity contacts perpendicular to the flow direction may divert flow more dramatically than asperity contacts parallel to the flow direction, i.e., Figure V.2(b). More importantly, flow may be funneled into localized regions if asperity contacts are arranged in the manner schematically similar to Figure V.2(c). Funneled flow in porous media has been indeed observed in the field (Kung, 1990ab). Thus, the ability of asperity contacts to divert flow depends on their correlation lengths, anisotropic ratio, and their orientation relative to the downward flow direction.

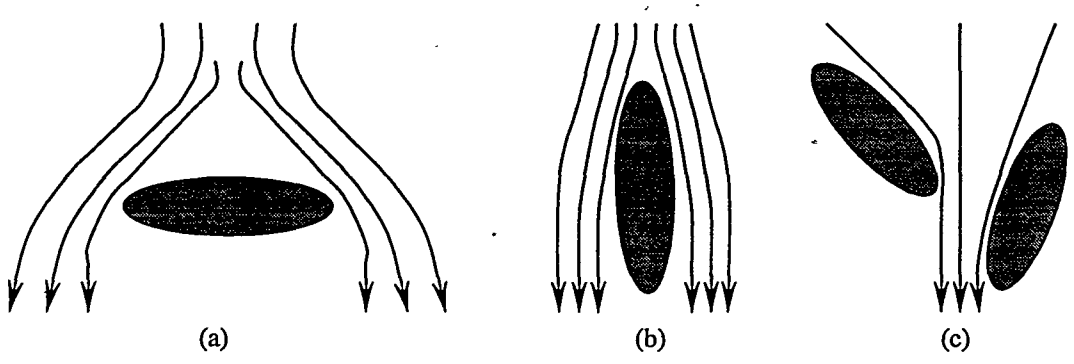


Figure V.2 Schematic representation of flow lines diverted by anisotropic asperity contacts which are represented as ellipses for simplicity.

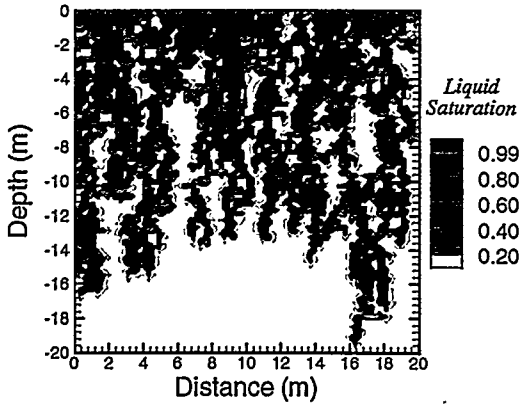
V.2 Impact of spatial discretization on liquid seepage

Recall that a sensitivity analysis of characterization accuracy with respect to spatial discretization was carried out in Chapter II. Results from that sensitivity analysis,

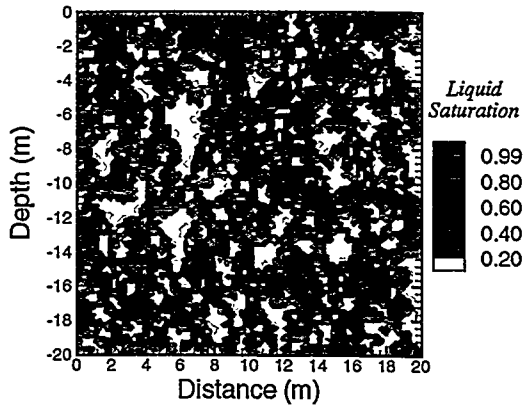
Figure II.8, indicated that the spatial variability of permeability fields could be adequately captured as long as $\Delta/\lambda_k < 0.5$. That sensitivity analysis is further extended in this section for investigating the effect of spatial discretization on flow simulations.

Simulated liquid saturations at breakthrough and steady state in synthetic fractures with simulated permeability fields shown in Figure III.8 are represented in Figures V.3 and V.4. The term "breakthrough" is simply defined as the condition when the first liquid reaches the depth of –19.9 m. The term "steady state" is herein defined as the point in simulation when the ratio of the total flux exiting the bottom boundary to the flux injecting to the top boundary exceeds 0.999. It should keep in mind that flow fields obtained by this convention are strictly pseudo steady-state flow fields.

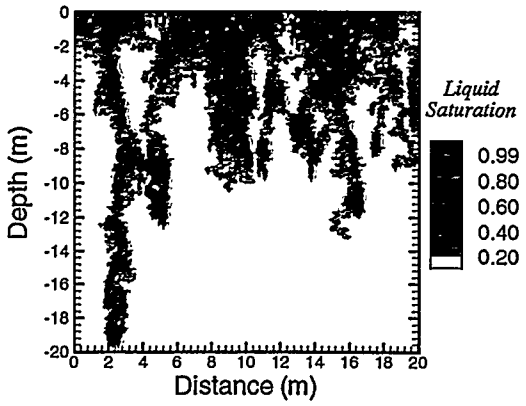
Consequently, fine spatial discretization does not appear to be necessary to model seepage through fractures with spatially correlated asperity contacts as long as $\Delta/\lambda_k < 0.5$. Figure V.3 shows that the discretization length (Δ) and correlation length (λ_k) both influence the simulated seepage pattern in a fracture with uncorrelated asperity contacts. In contrast, the simulated seepage patterns in a fracture with spatially correlated asperity contacts are remarkably similar. The similarity is independent of the spatial discretization as illustrated in Figures V.4(a) and V.4(b) (or Figures V.4(d) and V.4(e)) if the permeability fields have the same correlation lengths and the ratio Δ/λ_k is smaller than 0.5. This similarity is preserved in Figures V.4(c) and Figure V.4(f) even if the correlation length of permeability is increased. Thus, the seepage pattern is strongly affected by the correlation structure of asperity contacts but to a lesser extent the correlation structure of permeability, for $\Delta/\lambda_k < 0.5$.



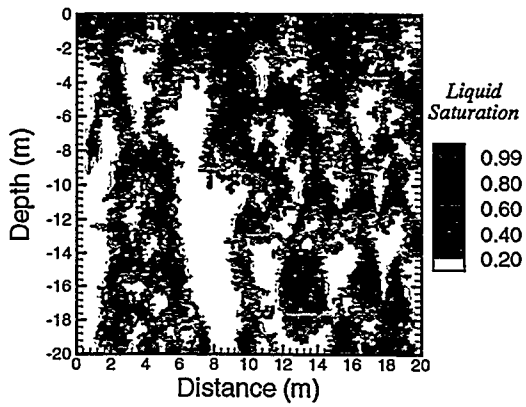
(a) $\Delta_x = 0.2$m, $\Delta_z = 0.2$m; $\lambda_{k,x}$=0.4m, $\lambda_{k,z}$=0.4m (Δ/λ_k=1/2)



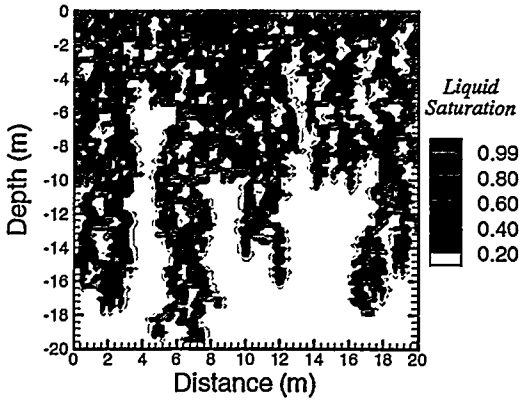
(d) $\Delta_x = 0.2$m, $\Delta_z = 0.2$m; $\lambda_{k,x}$=0.4m, $\lambda_{k,z}$=0.4m (Δ/λ_k=1/2)



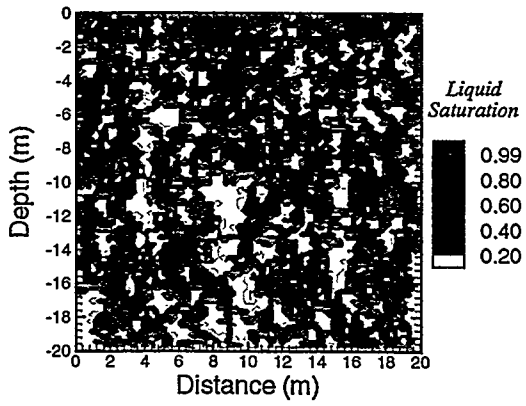
(b) $\Delta_x = 0.1$m, $\Delta_z = 0.1$m; $\lambda_{k,x}$=0.4m, $\lambda_{k,z}$=0.4m (Δ/λ_k=1/4)



(e) $\Delta_x = 0.1$m, $\Delta_z = 0.1$m; $\lambda_{k,x}$=0.4m, $\lambda_{k,z}$=0.4m (Δ/λ_k=1/4)

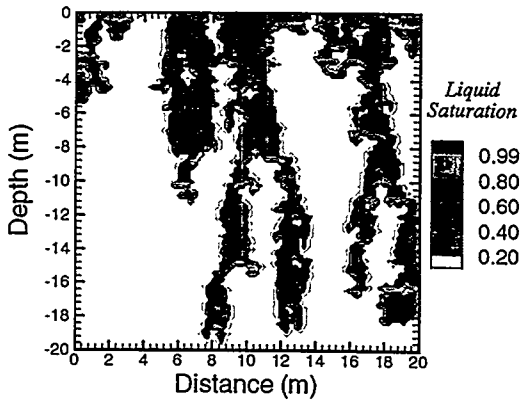


(c) $\Delta_x = 0.2$m, $\Delta_z = 0.2$m; $\lambda_{k,x}$=0.8m, $\lambda_{k,z}$=0.8m (Δ/λ_k=1/4)

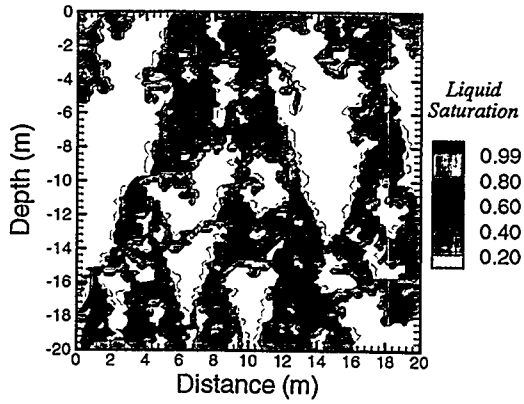


(f) $\Delta_x = 0.2$m, $\Delta_z = 0.2$m; $\lambda_{k,x}$=0.8m, $\lambda_{k,z}$=0.8m (Δ/λ_k=1/4)

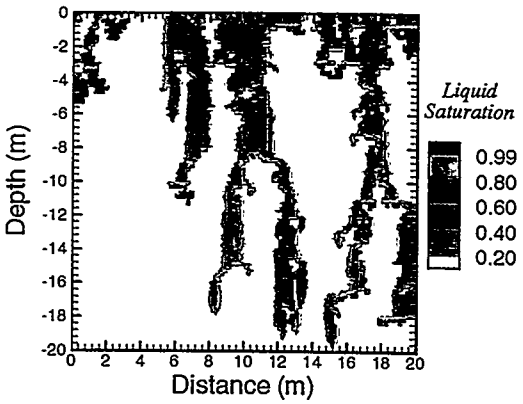
Figure V.3 Saturation at breakthrough (cases a to c) and steady state (cases d to f) in synthetic fractures shown in Figure III.8(a) to II.8(c), with isotropic, spatially random asperity contacts with different radii and grids with different values of Δ.



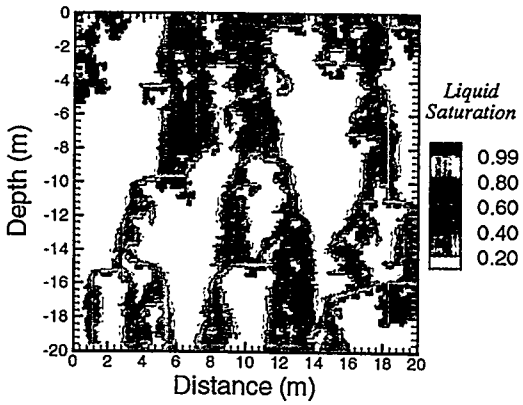
(a) $\Delta_x = 0.2$m, $\Delta_z = 0.2$m; $\lambda_{k,x}$=0.4m, $\lambda_{k,z}$=0.4m (Δ/λ_k=1/2)



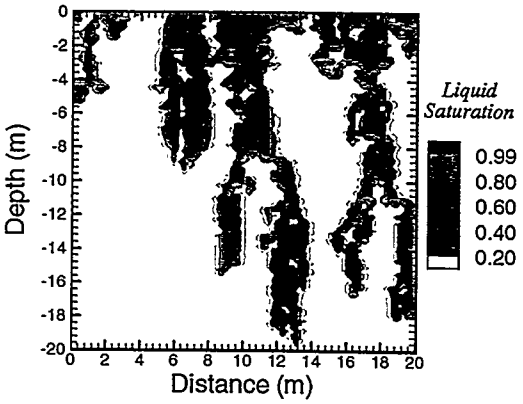
(d) $\Delta_x = 0.2$m, $\Delta_z = 0.2$m; $\lambda_{k,x}$=0.4m, $\lambda_{k,z}$=0.4m (Δ/λ_k=1/2)



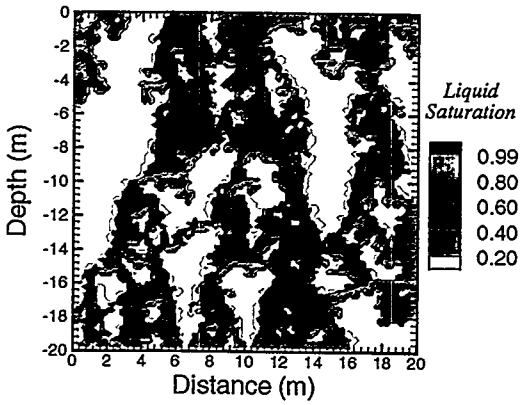
(b) $\Delta_x = 0.1$m, $\Delta_z = 0.1$m; $\lambda_{k,x}$=0.4m, $\lambda_{k,z}$=0.4m (Δ/λ_k=1/4)



(e) $\Delta_x = 0.1$m, $\Delta_z = 0.1$m; $\lambda_{k,x}$=0.4m, $\lambda_{k,z}$=0.4m (Δ/λ_k=1/4)



(c) $\Delta_x = 0.2$m, $\Delta_z = 0.2$m; $\lambda_{k,x}$=0.8m, $\lambda_{k,z}$=0.8m (Δ/λ_k=1/4)



(f) $\Delta_x = 0.2$m, $\Delta_z = 0.2$m; $\lambda_{k,x}$=0.8m, $\lambda_{k,z}$=0.8m (Δ/λ_k=1/4)

Figure V.4 Saturation at breakthrough (cases a to c) and steady state (cases d to f) in synthetic fractures shown in Figures III.8(d) to III.8(f), with isotropic, spatially correlated asperity contacts with different correlation lengths and grids with different values of Δ.

V.3 Seepage versus normal stress

Experimental data (Pyrak-Nolte *et al.*, 1987) has showed that contact areas in natural fractures increase with normal stress. However, the detailed mechanical properties of contact areas in fractures and their relationship with the normal stress are not interested here. Then, the relationship between liquid seepage and normal stress is simulated instead by changing the total fraction of asperity contacts. Four different fractions of asperity contacts are considered here, 15%, 25%, 35% and 40%. For each fraction of asperity contacts, twenty to thirty synthetic fractures were generated in order to obtain statistically homogeneous realizations of heterogeneous fractures.

Spatially random and spatially correlated asperity contacts corresponding to the different stress levels are shown in Figure V.5. The corresponding realizations of permeability fields obtained by conditioning on the asperity data are shown in Figure V.6 and V.7. The expected spatial correlation structure of the permeability fields is an isotropic exponential semi-variogram with nugget = 0.0, correlation length (λ_k) = 0.4 m, and sill number ranging from 90.0 to 190.0 in different realizations. Spatial discretization (Δ) is 0.2 m, i.e, the ratio $\Delta / \lambda_k = 0.5$.

Computed saturation at the time of breakthrough and steady state obtained using the different permeability fields is plotted in Figures V.8 to V.11. Flow simulations in fractures with spatially random asperity contacts and low normal stress exhibit numerous interconnected flow paths. As the normal stress increases, significant preferential flow occurs whether the asperity contacts are spatially random or spatially correlated. The preferential flow is accompanied by significant flow bypassing and ponding. The asperity

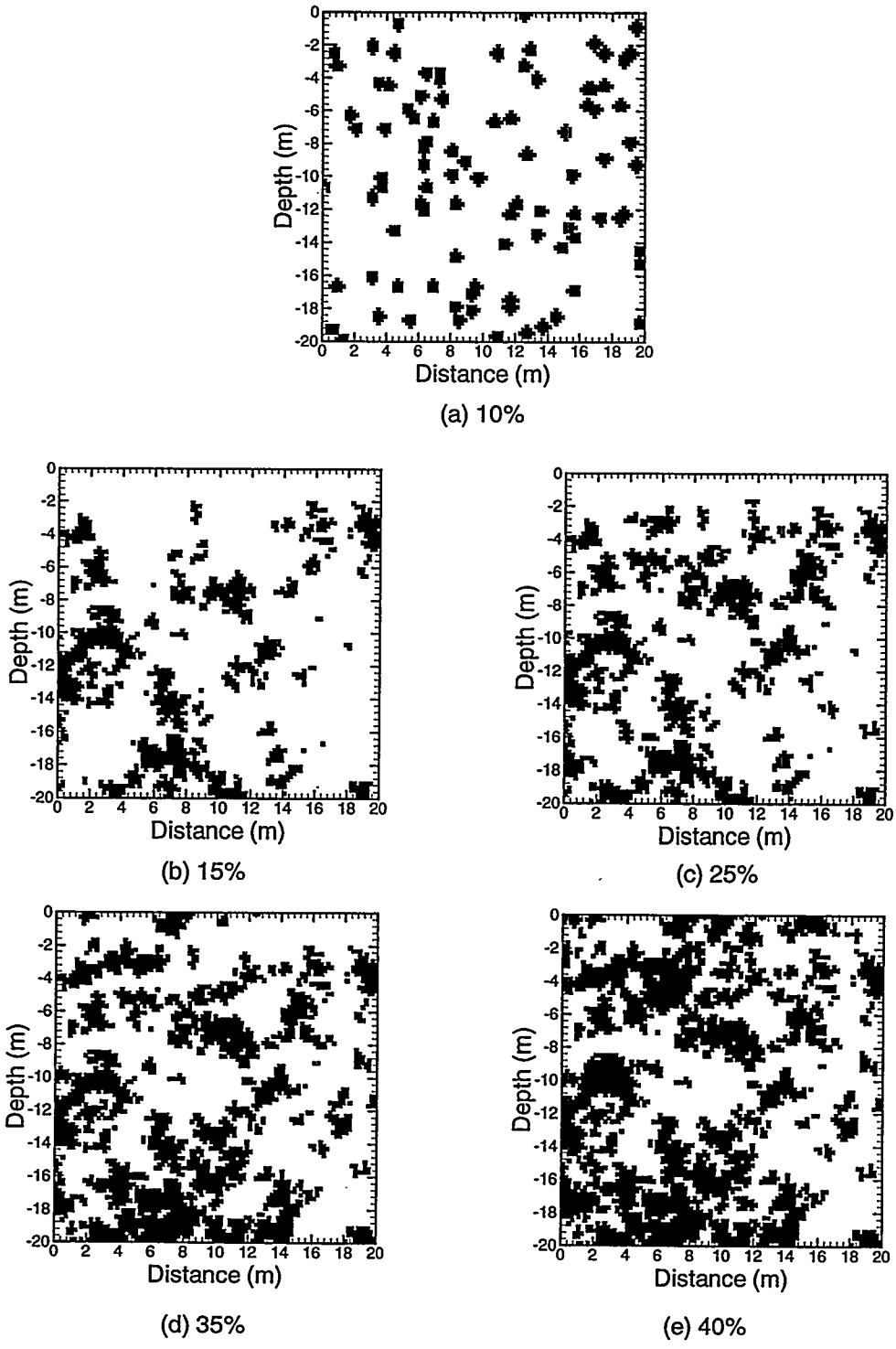
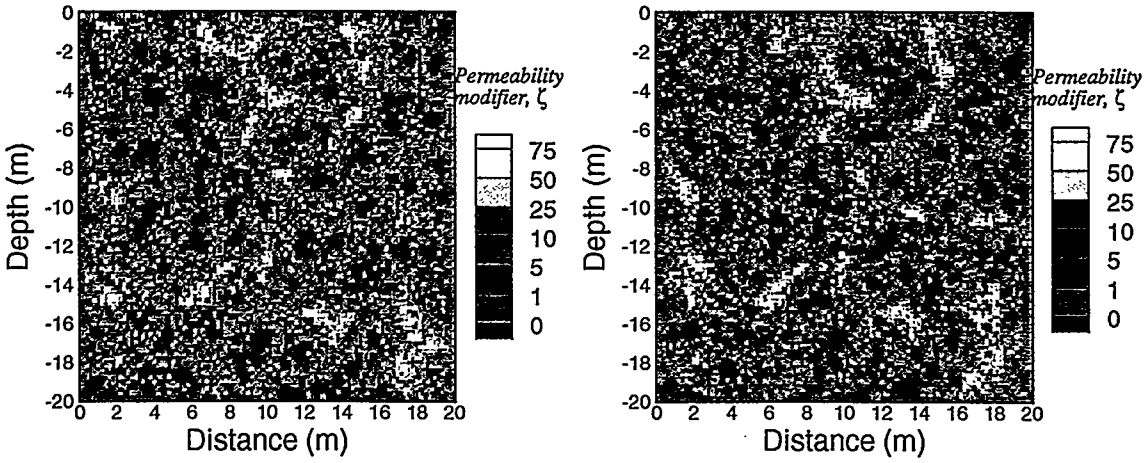
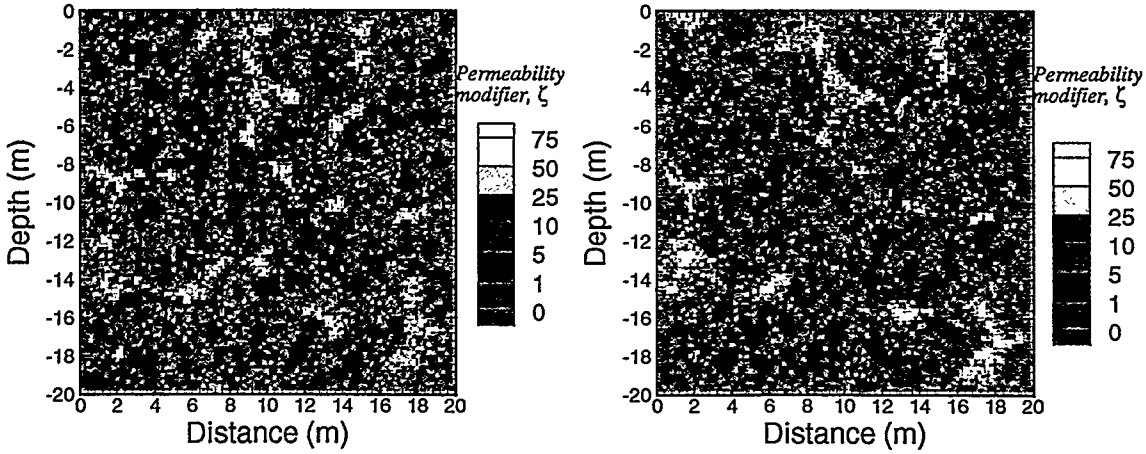


Figure V.5 Spatially random (case a) and spatially correlated asperity contacts (cases b to e), used as the conditioning data for heterogeneous fractures. Radius (r_0) and correlation length (λ_0) for asperity contacts are both 0.4 m. The correlation structure for cases b to d is an isotropic exponential semi-variogram with nugget = 0.0 and sill = 0.1875.



(a) permeability field with 15% spatially random asperity contacts

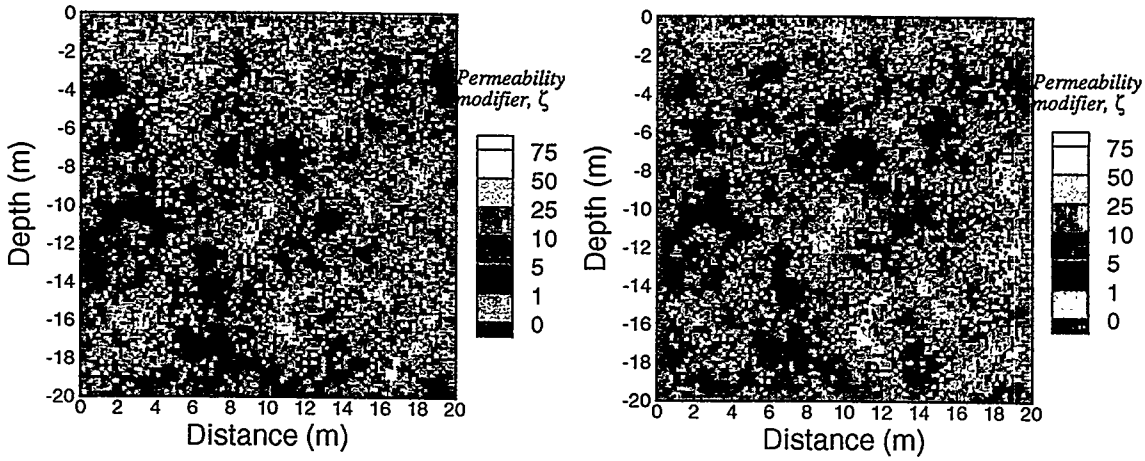
(b) permeability field with 25% spatially random asperity contacts



(c) permeability field with 35% spatially random asperity contacts

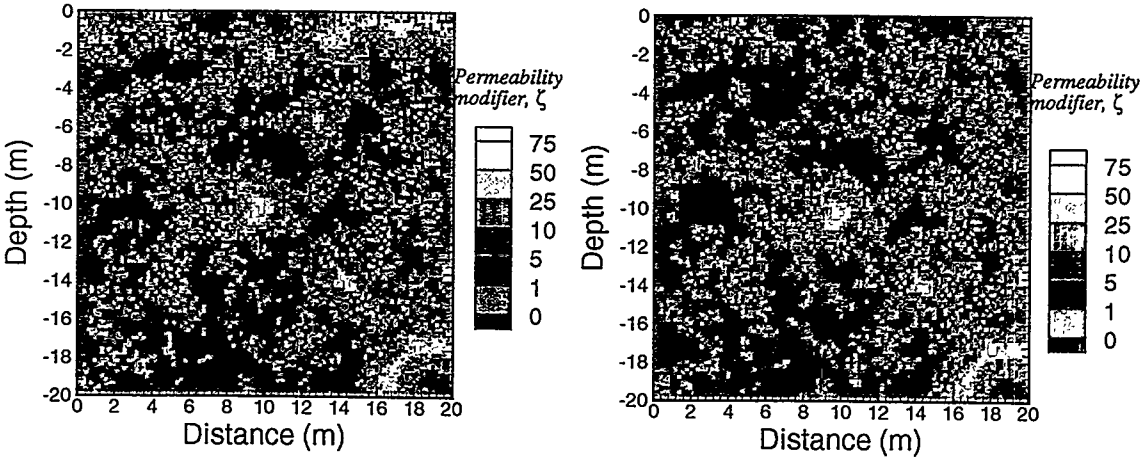
(d) permeability field with 40% spatially random asperity contacts

Figure V.6 Synthetic fractures conditioned on spatially random asperity contacts as shown in Figure V.5(a). The expected correlation structure is an isotropic semi-variogram with nugget = 0.0, sill = 190.0 and correlation length (λ_k)= 0.4m. Spatial discretization (Δ) = 0.2m.



(a) permeability field with 15% spatially correlated asperity contacts

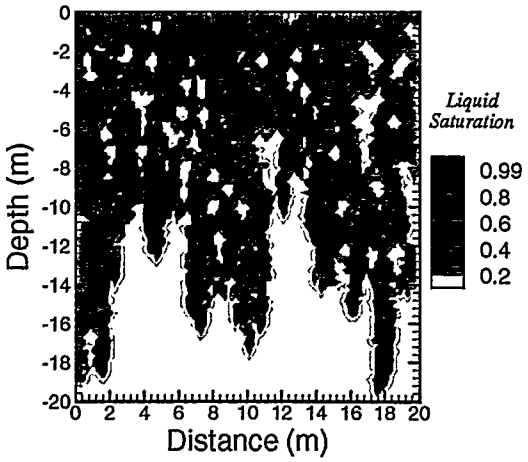
(b) permeability field with 25% spatially correlated asperity contacts



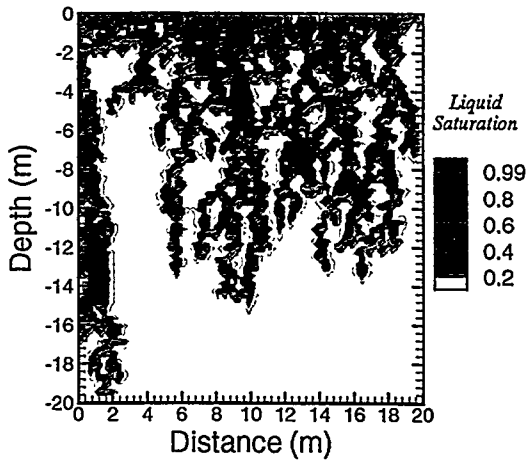
(c) permeability field with 35% spatially correlated asperity contacts

(d) permeability field with 40% spatially correlated asperity contacts

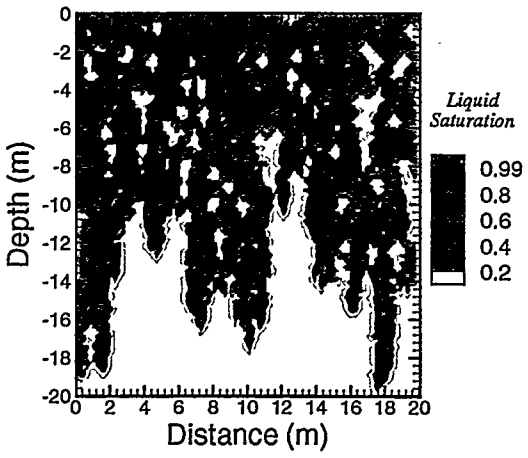
Figure V.7 Synthetic fractures conditioned on spatially correlated asperity contacts as shown in Figures V.5 (b) to V.5(e). The expected correlation structure is an isotropic semi-variogram with nugget = 0.0, and correlation length (λ_k) = 0.4m. Sill numbers are 120.0, 115.0, 100.0, and 90.0 for realizations (a), (b), (c) and (d), respectively. Spatial discretization (Δ) = 0.2m.



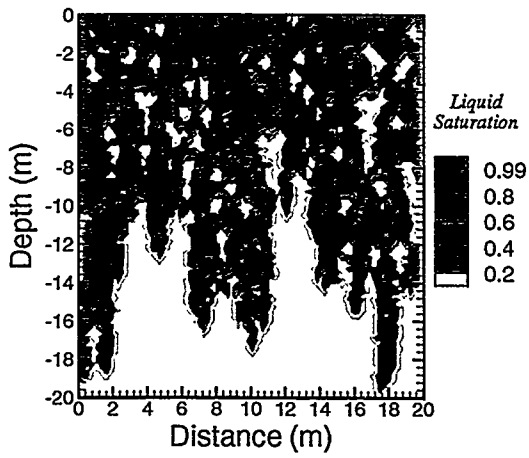
(a) 15% asperity contacts, Time = 54.92 hrs



(b) 25% asperity contacts, Time = 41.88 hrs

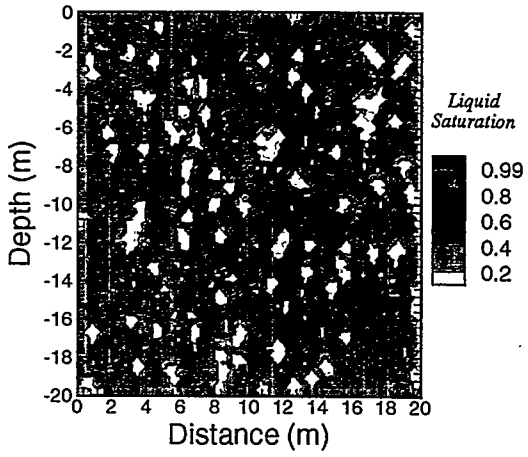


(c) 35% asperity contacts, Time = 37.31 hrs

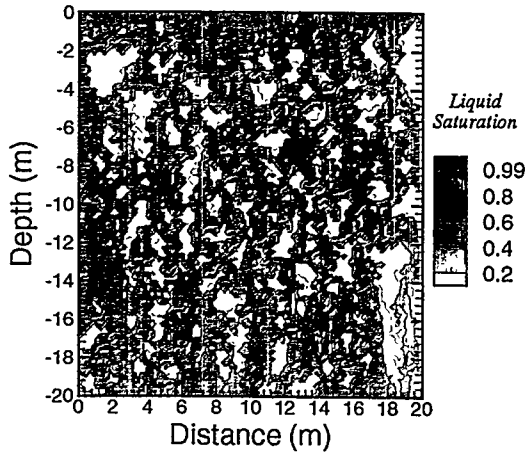


(d) 40% asperity contacts, Time = 34.85 hrs

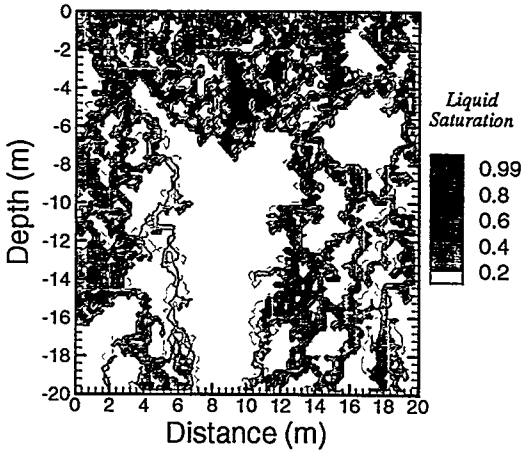
Figure V.8 Saturation at the time of breakthrough at the depth of -19.9 m, in fractures with spatially random asperity contacts. Initial water saturation is at the value of 0.15.



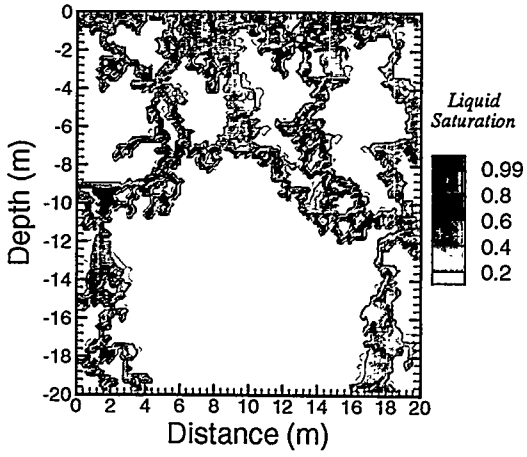
(a) 15% asperity contacts, Time = 13.01 days



(b) 25% asperity contacts, Time = 55.38 days

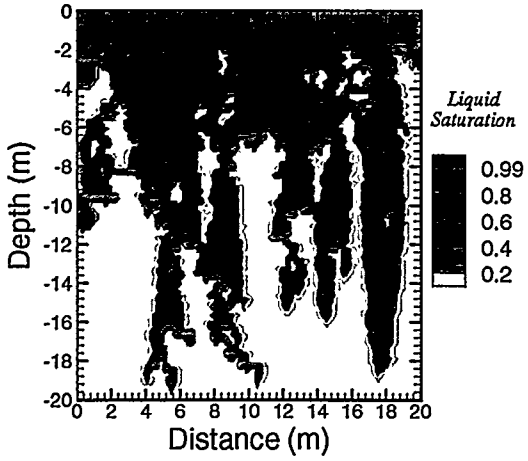


(c) 35% asperity contacts, Time = 115.95 days

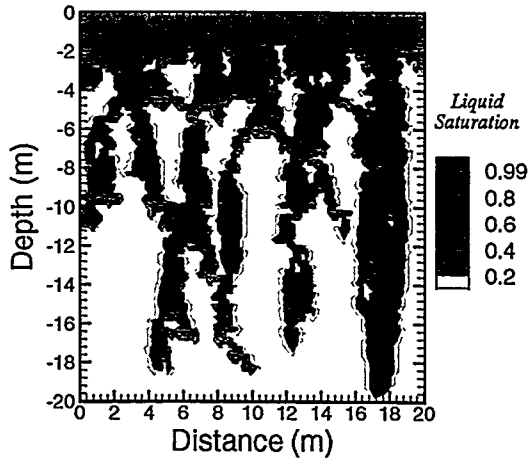


(d) 40% asperity contacts, Time = 18.80 days

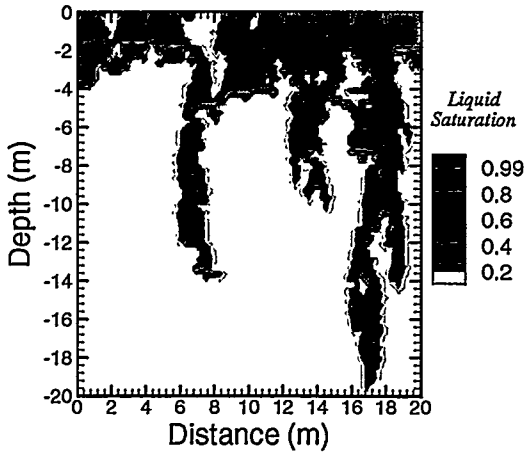
Figure V.9 Steady state saturation in fractures with spatially random asperity contacts. Initial water saturation is at the value of 0.15.



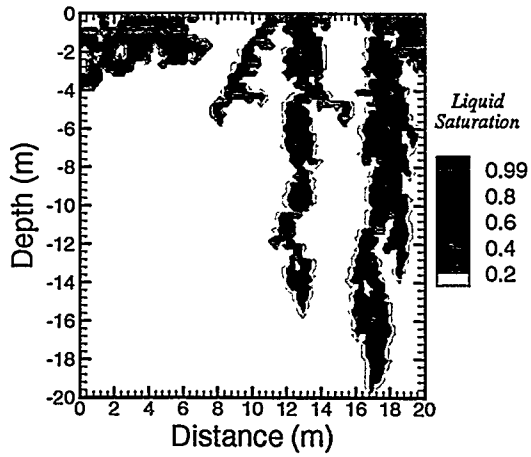
(a) 15% asperity contacts, Time = 53.69 hrs



(b) 25% asperity contacts, Time = 49.20 hrs

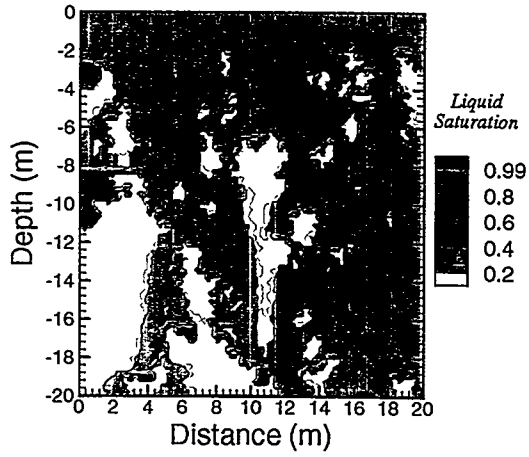


(c) 35% asperity contacts, Time = 34.33 hrs

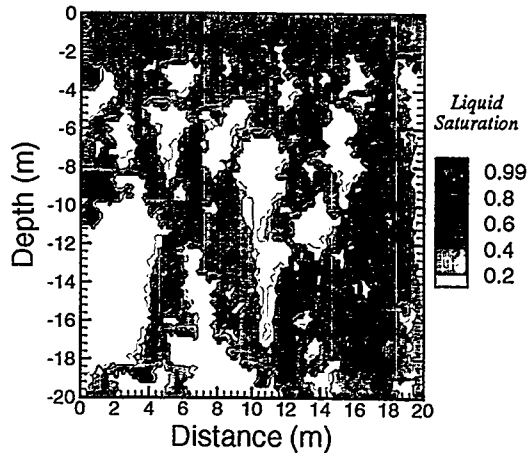


(d) 40% asperity contacts, Time = 30.55 hrs

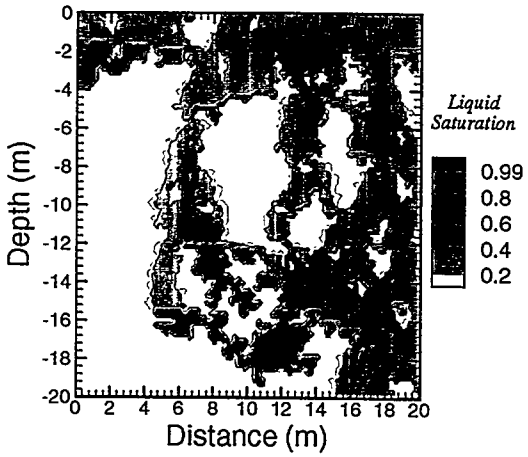
Figure V.10 Saturation at the time of breakthrough at the depth of -19.9 m, in fractures with spatially correlated asperity contacts. Initial water saturation is at the value of 0.15.



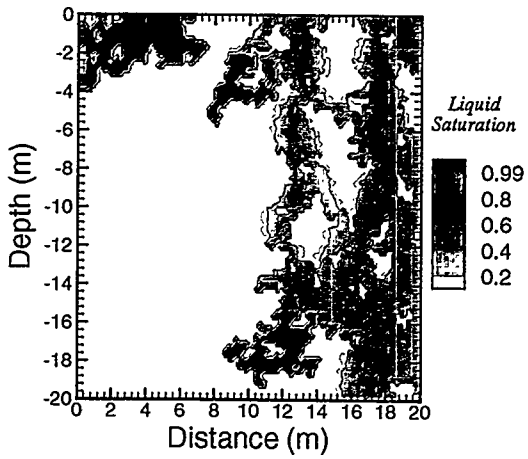
(a) 15% asperity contacts, Time = 13.01 days



(b) 25% asperity contacts, Time = 55.38 days



(c) 35% asperity contacts, Time = 115.95 days



(d) 40% asperity contacts, Time = 18.80 days

Figure V.11 Steady state saturation in fractures with spatially correlated asperity contacts. Initial water saturation is at the value of 0.15.

contacts divert the water into spatially localized fingers that may further proceed downward, merge with other fingers, or be terminated/ponded on laterally extensive sub-horizontal asperity contacts.

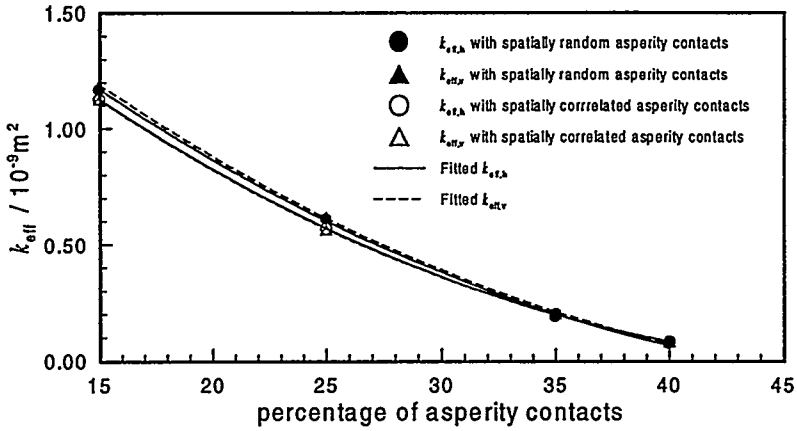


Figure V.12 Arithmetic means of effective permeabilities in synthetic fractures with spatially random and spatially correlated asperity contacts.

Horizontal and vertical effective permeabilities for each of the synthetic fractures were calculated for each fraction of asperity contacts. To calculate effective permeability, single phase flow simulations were performed to steady state and the permeability was calculated based on Darcy's law (see Appendix D for details). The results are plotted in Figure V.12 and a monotonically decreasing trend with normal stress as would be expected.

The vertical advance of the fastest finger with respect to time is plotted in Figure V.13 for flow simulations with spatially correlated asperity contacts. The slopes of these curves tend to increase with increasing normal stress, suggesting faster breakthrough in fractures subject to increasing normal stresses. The occurrence of faster breakthrough in fractures results from the increasing degree of preferential flow as normal stress increases, along with the increased Darcy's velocity as the flow funnels into localized

fingers. The trend of shorter breakthrough time is further verified by histograms of breakthrough time shown in Figure V.14. The vertical advance curves for fractures subject to low normal stress tend to be linear because the effects of bypassing and ponding are weak. However, the curves become irregular as flow bypassing and ponding become significant at high normal stress.

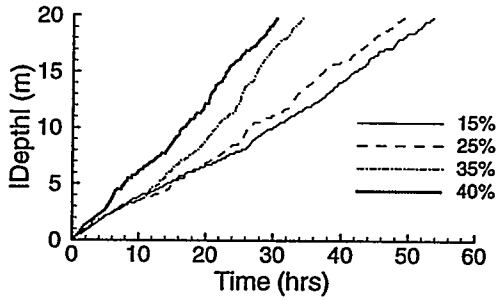


Figure V.13 Vertical advance curves obtained from flow simulations with spatially correlated fracture asperities.

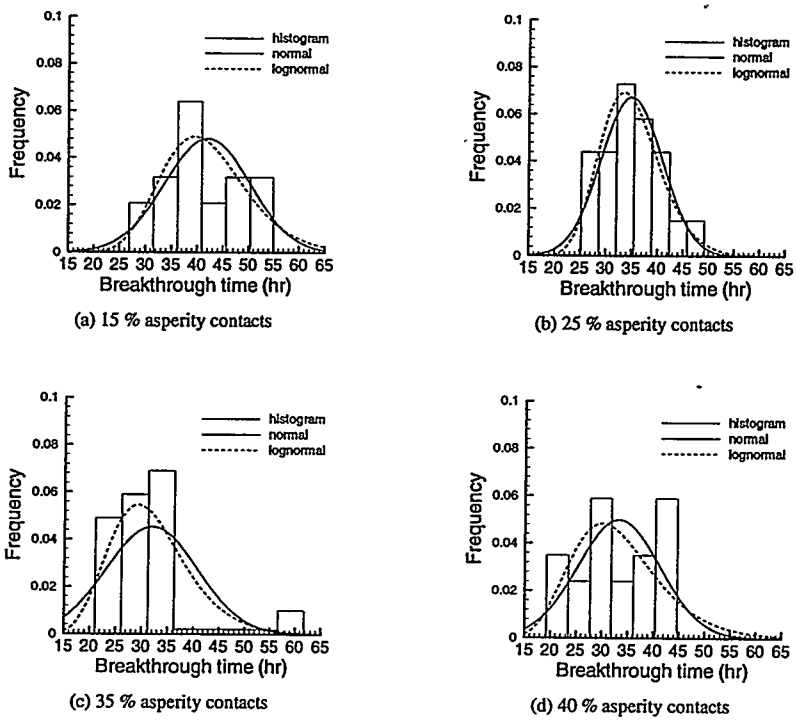
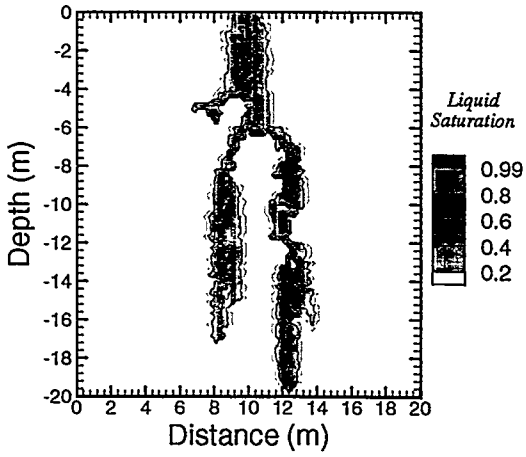


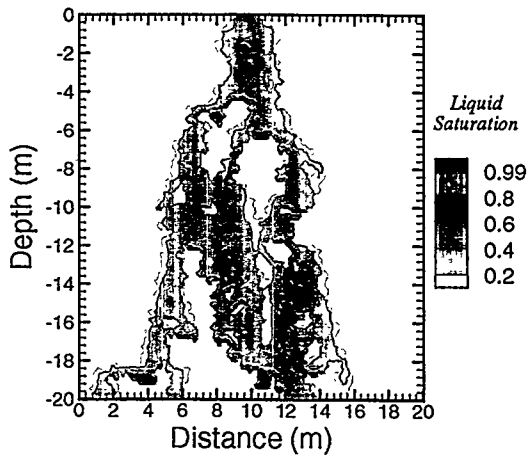
Figure V.14 Histograms of time to breakthrough in fractures subject to increasing normal stress.

An alternative boundary condition is to consider localized injection. Figure V.15 shows the transient and steady state flow fields in fractures with spatially correlated asperity contacts, subject to localized injection. Water is injected at the center of the top boundary. The injection zone has a lateral extent of 1.0 m,. No flux is allowed to across the top boundary outside the injection zone. The constant injection rate is 10^{-3} kg/s, and initial conditions and other boundary conditions are the same as before. Seepage patterns similar to those for distributed injection cases are observed in these simulations, such as fingering, ponding, bypassing, and lateral spreading. However, the preferential flow paths for localized injection cases are obviously different from those for the distributed injection cases. In addition, the location of the first breakthrough for the localized injection case may or may not be the same as the distributed injection case. These flow phenomena were also observed by Pruess (1998) in his simulations using synthetic fractures with spatially random asperity contacts. Furthermore, he observed that seepage patterns in natural fractures strongly depends on fracture permeability, capillary effects, and applied flow rate. Thus, it is expected that these observations also apply to the simulation results in the present study.

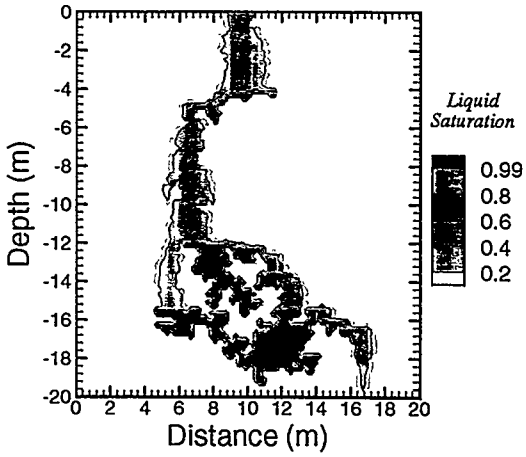
A vertical advance curve records the downward migration of the fastest finger. Thus, an abrupt change in the slope of the curve indicates the emergence of a faster finger or acceleration/retardation of seepage. Usually, the change of slope is ascribed to ponding on asperities. For example, the curve in Figure V.16(a) exhibits an arrest in the fluid advance before breakthrough. The corresponding flow simulation in Figure V.16(b) shows that this is indeed because of ponding at about –15 m. In addition, the slope of the advance curve after ponding may increase or decrease. For example, Figure V.16(b)



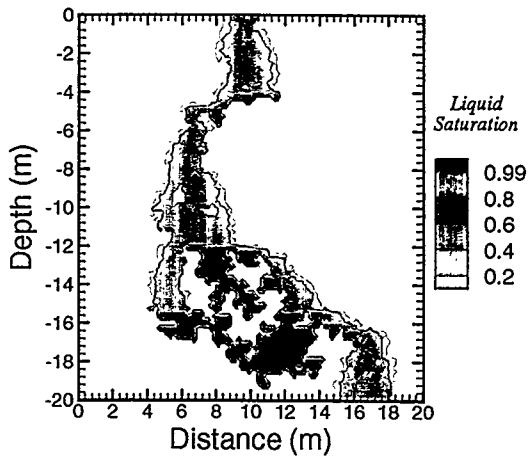
(a) 25% asperity contacts
Transient, time =11.79 hrs



(b) 25% asperity contacts
Steady state, time = 27.42 days

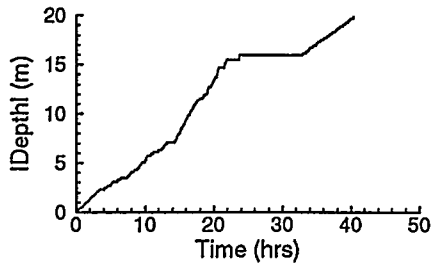


(c) 35% asperity contacts
Transient, time = 25.18 hrs

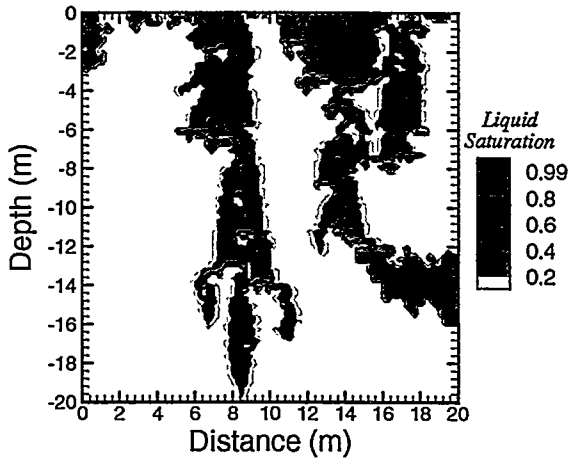


(d) 35% asperity contacts
Steady state, time = 12.52 days

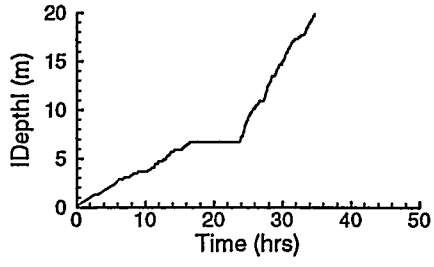
Figure V.15 Transient and steady state saturation with localized injection in fractures with spatially correlated asperity contacts.



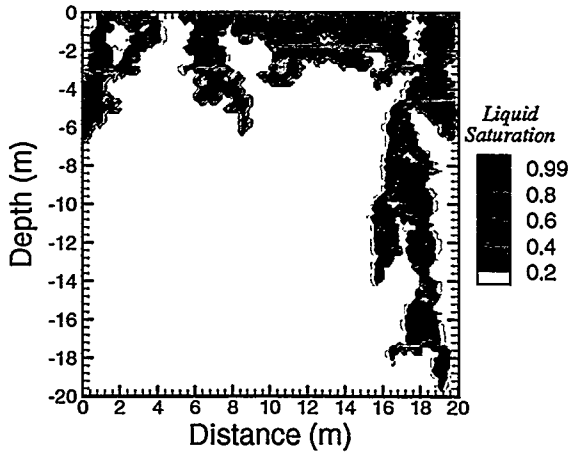
(a) Vertical advance curve in fractures with 25% spatially correlated asperity contacts



(b) Saturation corresponding to curve (a) Breakthrough time = 40.39 hrs



(c) Vertical advance curve in fractures with 35% spatially correlated asperity contacts



(d) Saturation corresponding to curve (c) Breakthrough time = 34.71 hrs

Figure V.16 Vertical advance curves and corresponding saturation in synthetic fractures with spatially correlated asperity contacts. Figure IV.16(a) and V.16(c) feature the effects of ponding and flow funneling, respectively.

shows that finger on the left of the ponded zone breaks through the bottom boundary. This finger has lower seepage velocity and the curve in Figure V.16(a) shows a flatter slope just before breakthrough. In contrast, the curve in Figure V.16(c) shows that seepage starts with a smaller velocity, becomes arrested by ponding for approximately 8 hrs, and then proceeds faster toward the bottom boundary. The ponding is clearly visible in Figure V.16(d).

Breakthrough curves (BTCs) corresponding to flow simulations in Figure V.11 are presented in Figures V.13(b). For better representation, tails of BTCs after 10 days are truncated because they are essentially asymptotes approaching to flux ratio at 1.0.

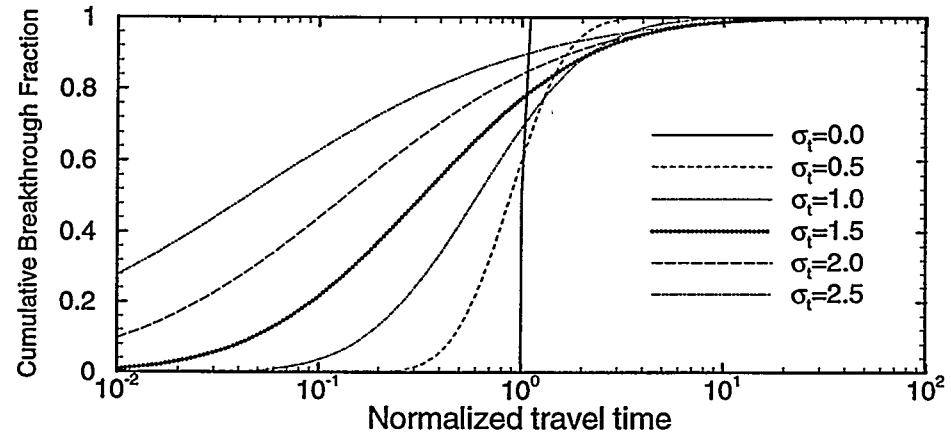
Chesnut (1992, 1994) suggested that travel time breakthrough curves of groundwater through unsaturated media can be approximated by a log-normal model. He proposed that the cumulative distribution function for groundwater travel time may be written as

$$P(t \le t_0) = \Phi\left(\frac{\ln(t_0 / < t >) + \sigma_t^2}{\sigma_t}\right) \tag{5.3}$$

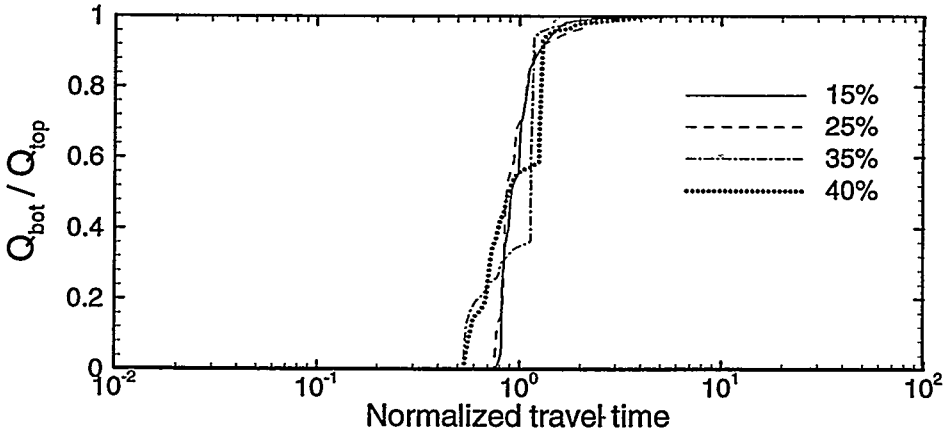
Here, $P(t \le t_0)$ is the fraction of fluid flowing between inflow and outflow boundaries for which travel time is less than or equal to t_0, $<t>$ is the mean travel time, σ_t is the standard deviation of the natural logarithm of travel time, and Φ is the cumulative distribution function of a normal distribution, i.e.,

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-u^2/2\right) du = \frac{1}{2} \operatorname{erfc}\left(-x/\sqrt{2}\right) \tag{5.4}$$

in which erfc() is the complementary error function. An important feature of the log-normal model given in Eq(5.3) is that a significant fraction of flow has travel times much shorter than the mean travel time. This effect becomes obvious for strongly heterogeneous media (larger σ_t), as shown in Figure V.17(a). Note that the horizontal axis of Figure V.17(a) is the travel time normalized by the mean travel time, $<t>$, that is fixed at 1.0.



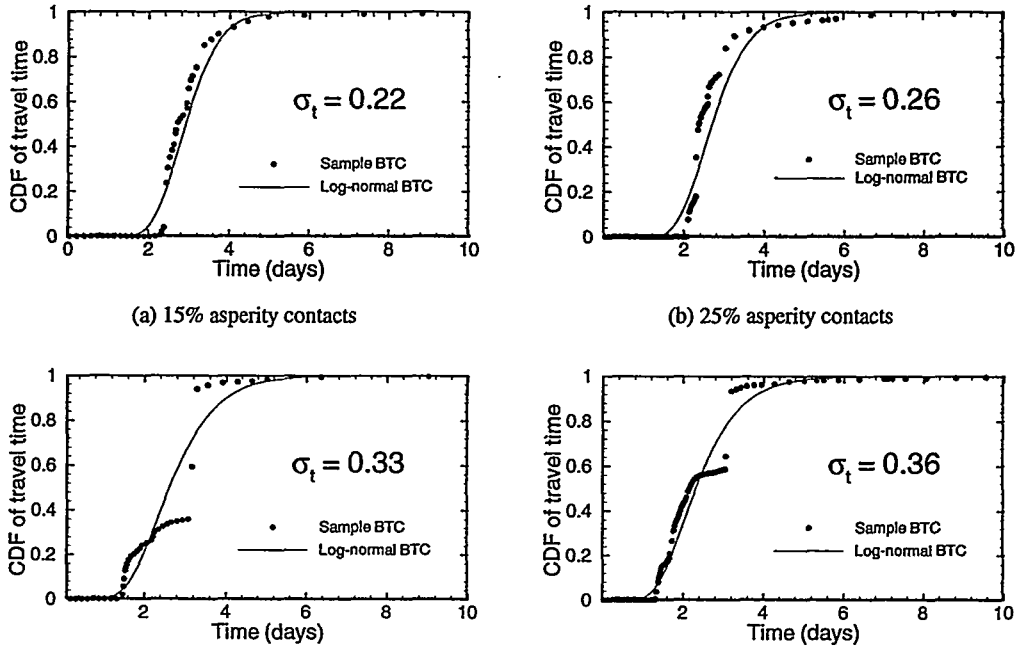
(a) Chesnut's model of log-normal BTC's with different values of σ_t



(b) Realized BTC's in fractures with spatially correlated asperity contacts

Figure V.17 (a) Theoretical log-normal travel time model for different values of the heterogeneity parameter, σ_t (Chesnut, 1992). The mean travel time, $<t>$, is fixed at 1.0 for all curves, which is also the normalizing factor for the horizontal axis; (b) Simulated BTCs for flow simulations in fractures with spatially correlated asperity contacts. Note that the horizontal axis is the travel time normalized by the mean travel time $<t>$.

The log-normal model is fitted to each of curves shown in Figure V.17(b), which are the BTCs for flow simulations in fractures with spatially correlated asperity contacts. The fitting begins by interpolating the sample BTC data and then taking numerical derivative of the interpolated BTC. These numerical derivatives then serve as the approximated probability distribution of the BTC data, from which the mean and variance of travel time can be estimated. Subsequently, analytical pdf and CDF can be obtained based on these two parameters. The fitted results in Figure V.18 show that the log-normal model does not adequately fit the individual travel time data. The reason for this poor fit is that water transport is strongly affected by the spatial distribution of permeability and the associated effects such as ponding, bypassing, and change of seepage velocity. Thus, the log-normal model is too simple to accurately capture those complicated flow effects.



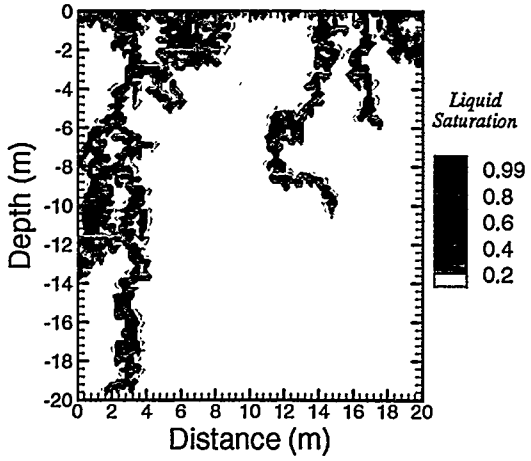
(a) 15% asperity contacts

(b) 25% asperity contacts

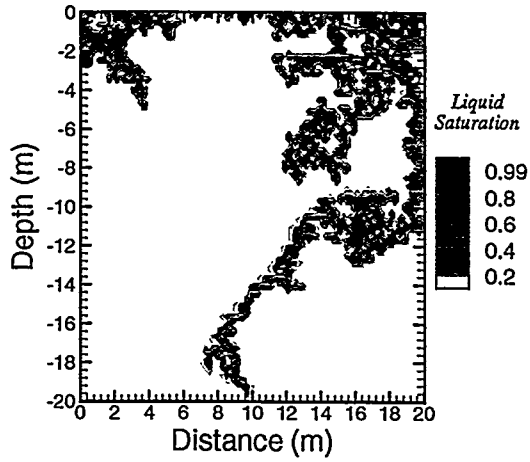
(c) 35% asperity contacts

(d) 40% asperity contacts

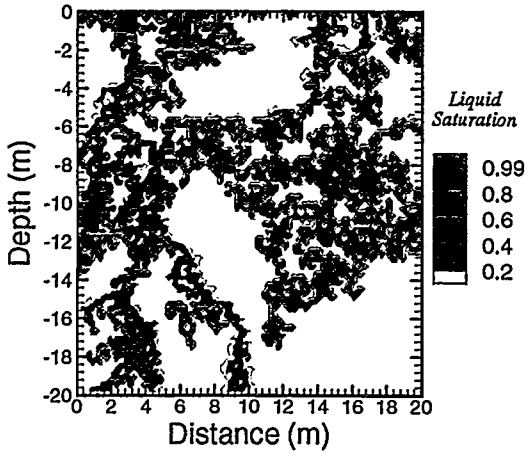
Figure V.18 Results of fitting the log-normal travel time model to each of the BTCs shown in Figure V.17(b).



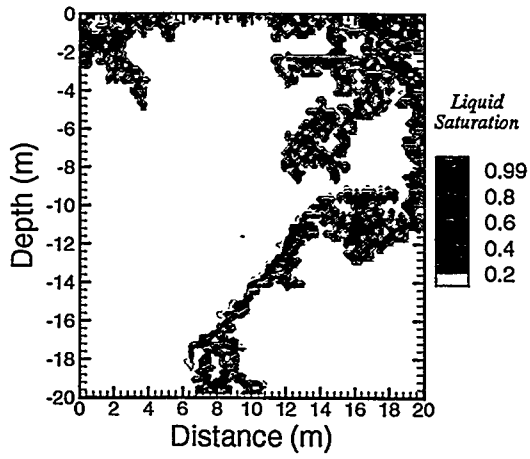
(a) At breakthrough, time = 28.08 hrs



(c) At breakthrough, time = 43.56 hrs



(b) At steady state, time = 73.30 days



(d) At steady state, time = 7.39 days

Figure V.19 Saturation at breakthrough and steady state in fractures subject to high normal stress, i.e., the total fractions of asperity contacts is 40%. These results illustrate the effect of seepage retardation (cases a and b) and acceleration (cases c and d) due to ponding.

Breakthrough curves in fractures subject to low normal stress tend to be smooth and monotonically increasing curves. This is because the effect of asperity contacts is not significant and liquid gradually comes out the entire bottom boundary in a spatially uniform manner. For fractures subject to high normal stress, however, BTCs behave like a step function because only a number of fingers can break through the bottom boundary. In addition, seepage velocity within these localized fingers is generally faster, resulting in the fast increase of a BTC. Moreover, such BTCs may intermittently exhibit horizontal segments, which are generally the result of impedance by ponding. For example, the effect of seepage impedance and acceleration can be demonstrated respectively from realizations in Figures V.19(b) and V.19(d); and the solid and dashed BTCs in Figure V.20, respectively. BTCs in fractures subject to intermediate normal stress behave intermediately between the two extremes, and their shapes depend on the heterogeneity of fractures, see the dash-dotted BTC in Figure V.20.

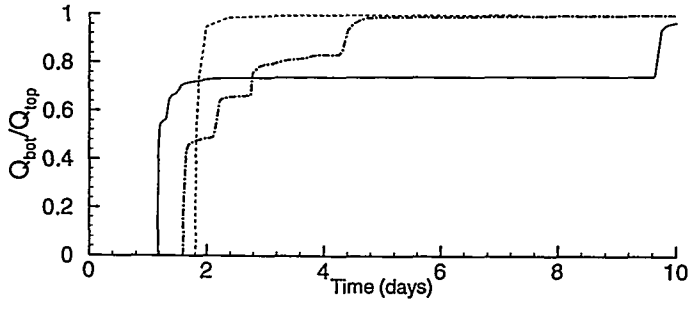
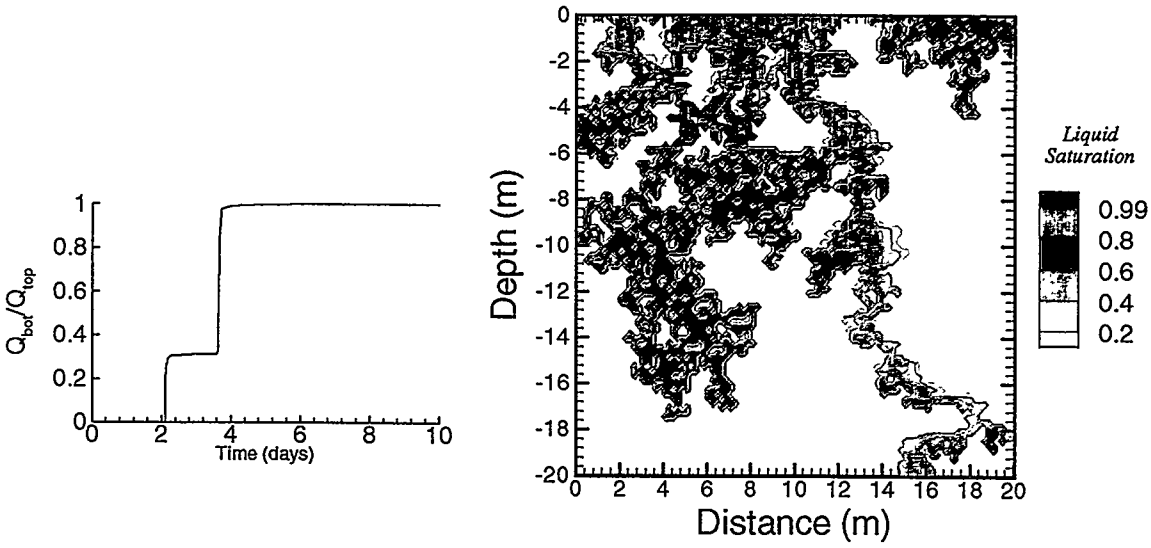


Figure V.20 BTCs showing the effects of seepage impedance (the solid curve) and seepage acceleration (the dashed curve), corresponding to flow realizations in Figures V.19(b) and V.19(d), respectively. The dash-dotted BTC shows the interchanging effects of seepage impedance and acceleration.

The mechanisms of seepage acceleration and impedance as a result of ponding in fractures subject to high normal stress are not independent but compete with each other.

A vertical line shown in a BTC simply means that seepage after breakthrough is dominated by the gravity effect. Competition between gravity and ponding will not start until other fingers also reach the bottom boundary. If the gravity effect still dominates, discharge at the bottom boundary would keep growing and the BTC would remain to be vertical. If, however, seepage is impeded by ponding, discharge would keep steady for a certain time during which the BTC would show a horizontal segment. Not until the gravity effect becomes significant again would the BTC show an abrupt increase in slope. For example, see the BTC in Figure V.21(a) and the corresponding distribution of saturation. In general, the competition between gravity and ponding is usually intermittent; for example, the dash-dotted BTC in Figure V.20.

The quantitative relationship between the degree of ponding and the total fraction of asperity contacts deserves further investigation. The degree of ponding is defined as follows. First, ponding regions are defined as the wetted region, in either transient or steady state flow fields, in which water saturation is one. Here the term "wetted region" is defined as the total volume in which water saturation is greater than the residual water saturation. Numerically, the cutoff saturations for ponded and wetted regions are chosen as 0.999 and 0.151, respectively. The degree of ponding can then be defined as the volumetric ratio of the ponded regions to the wetted region. Various factors may affect the total volume of ponded regions in heterogeneous fractures, including the total fraction of asperity contacts, correlation length of asperity contacts and permeability, and the correlation directions of asperity contacts and permeability. The parameter of the total fraction of asperity contacts is considered in this section because only synthetic fractures with isotropic asperity contacts and permeability are considered.



(a) BTC

(b) Saturation at steady state

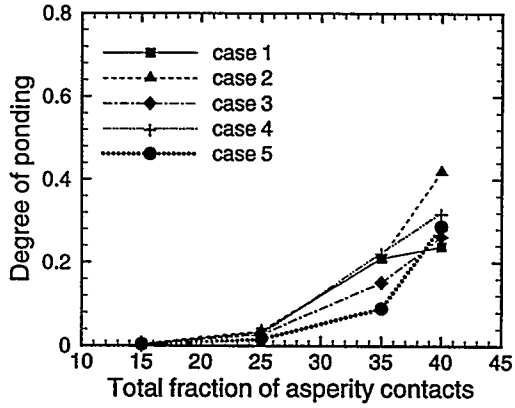
Figure V.21 Breakthrough curve and saturation at steady state in a synthetic fracture subject to high normal stress. The competition between gravity and ponding effects can be seen from the BTC.

Figure V.22 illustrates the degree of ponding in synthetic fractures with spatially random or spatially correlated asperity contacts, for transient as well as steady state flow fields. The plots in Figure V.22 suggests that the degree of ponding generally increases with the total fraction of asperity contacts, irrespective of the spatial correlation of asperity contacts and the status of the flow field.

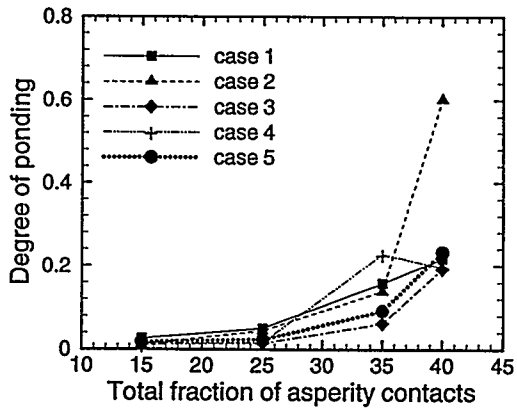
V. 4 Seepage dispersion and its dependence on asperity contacts

Most BTCs in this study exhibit a non-uniformly increasing trend with respect to time. In this study, seepage dispersion is the term for describing the spreading behavior of liquid seepage in heterogeneous fractures. It can be quantitatively measured by a coefficient (D_s) which is the ratio of ($t_{90} - t_{10}$) to t_{50}, i.e., $D_s = (t_{90}-t_{10})/t_{50}$. The parameters t_{90}, t_{50} and t_{10} are the travel times at which the flux ratios are 90%, 50% and 10% of the steady state flux (Neretnieks, *et al.*, 1982), respectively.

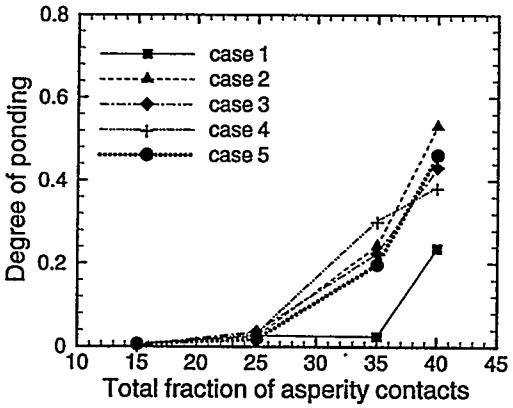
Histograms of t_{10}, t_{50}, and t_{90} are plotted in Figures V.23 and V.24 for synthetic fractures with spatially random or spatially correlated asperity contacts, respectively. Each histogram is fitted by a normal and a log-normal distributions. In general, the log-normal distribution usually fits the travel time data better than the normal distribution. The trend for t_{10} behaves as the breakthrough time, i.e., t_{10} generally decreases with increasing normal stress. Similarly, t_{50} also decreases with increasing normal stress. The travel time t_{90} approximately follows the trend of steady state time, i.e., t_{90} tends to be smaller for low and high normal stresses but larger for intermediate normal stress.



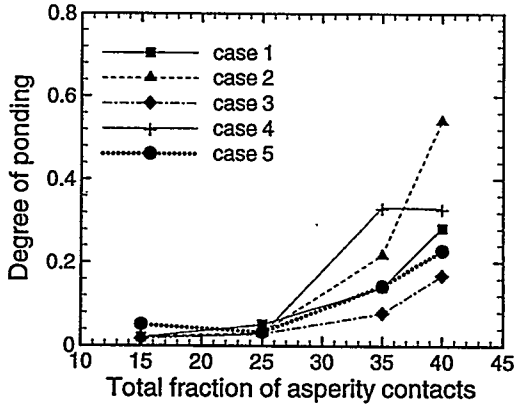
(a) Transient flow fields with spatially random asperity contacs



(c) Transient flow fields with spatially correlated asperity contacts



(b) Steady state flow fields with spatially random asperity contacts



(d) Steady state flow fields with spatially corelated asperity contacts

Figure V.22 The degree of ponding in isotropic fractures with isotropic, spatially random and spatially correlated asperity contacts, for transient and steady state flow fields.

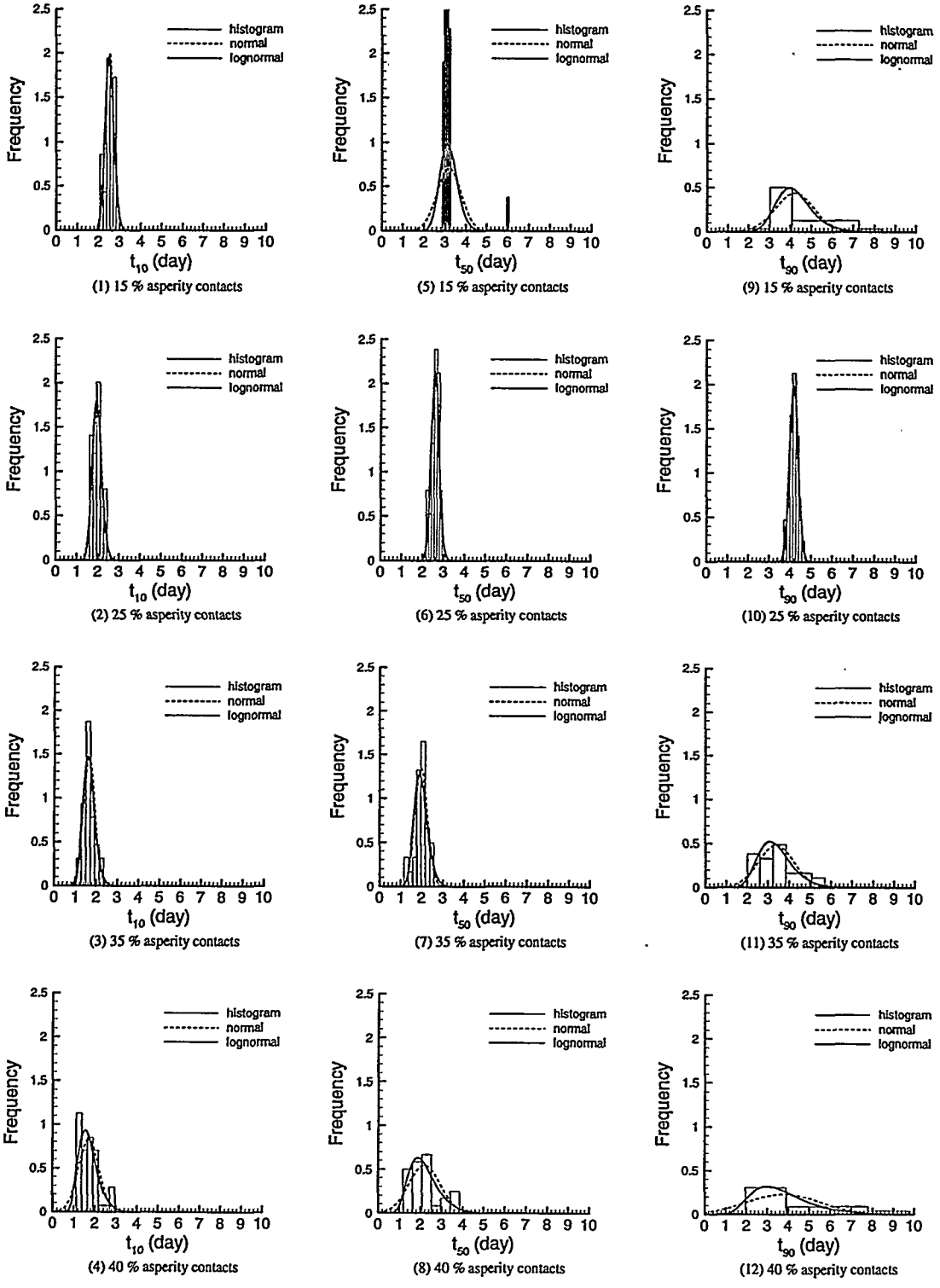


Figure V.23 Histograms of t_{10}, t_{50}, and t_{90} for fractures with spatially random asperity contacts.

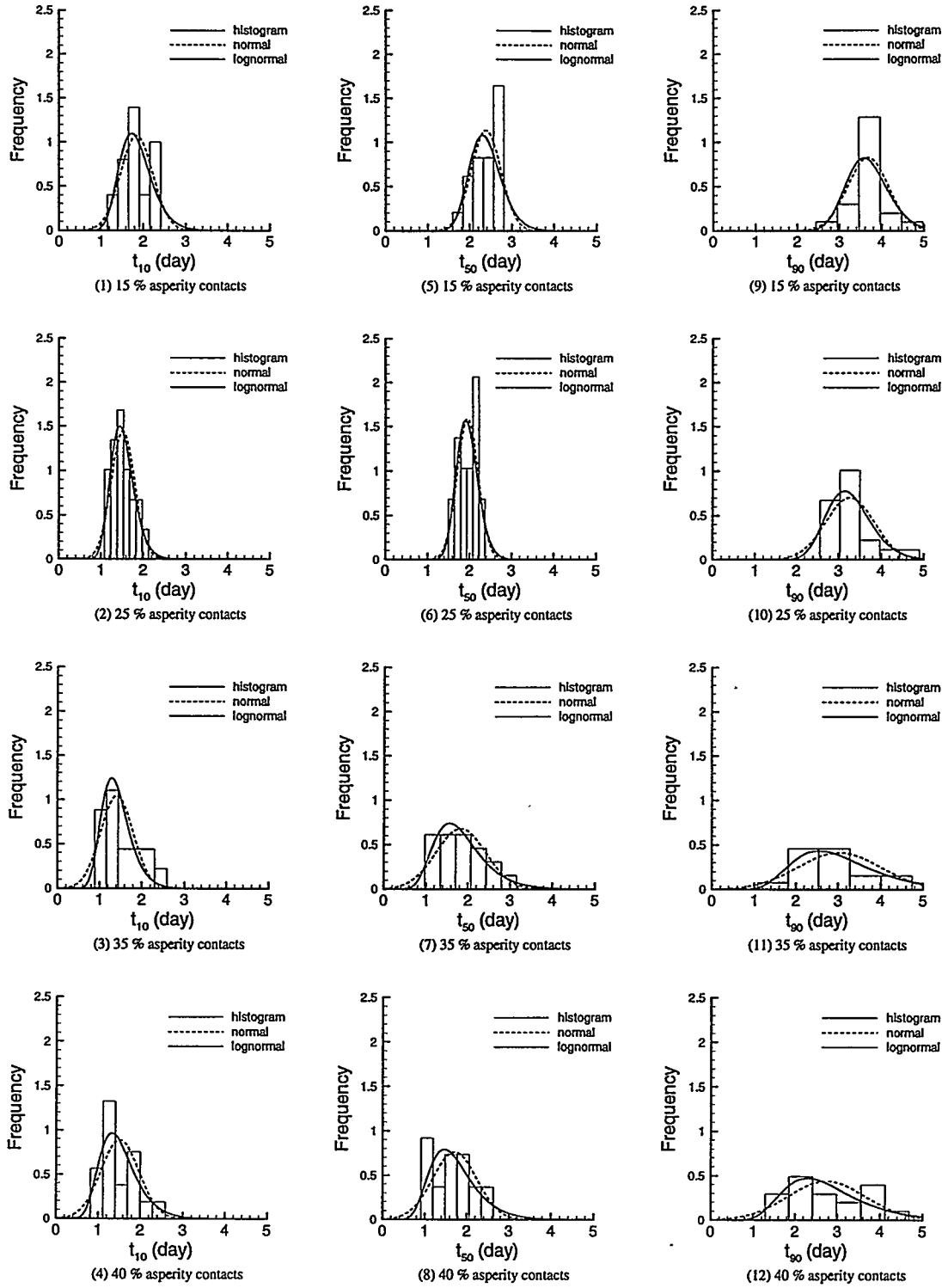


Figure V.24 Histograms of t_{10}, t_{50}, and t_{90} for fractures with spatially correlated asperity contacts.

Figure V.25 shows the histograms of the coefficient of seepage dispersion (D_s) for fractures with spatially random and spatially correlated asperity contacts. Note that the horizontal axes of histograms with 40% asperity contacts begin from –1. This is for representation purpose only because the fitted log-normal distribution is highly right-skewed toward zero. Statistics for fitted normal and log-normal distributions are listed in Table V.1, along with the goodness of fit data (p-value) obtained by the chi-square test. The quantitative evidence that the log-normal distribution fits the sample data better than the normal distribution is reflected by the larger p-value for the log-normal distribution (Table V.1). Thus, each histogram shown in Figure V.25 is fitted by a log-normal distribution.

Table V.1. Statistics of normal and log-normal distributions fitted to the sample data of D_s in fractures with spatially random and spatially correlated asperity contacts.

	Normal distribution				Log-normal distribution			
Fraction of asperity contacts	$\overline{D_S}$	s_{D_s}	$s_{D_s}/\overline{D_s}$	p-value	$\overline{\ln(D_s)}$	$s_{\ln(D_s)}$	$\frac{s_{\ln(D_s)}}{\overline{\ln(D_s)}}$	p-value
	Fractures with spatially random asperity contacts							
15%	0.530	0.143	0.269	0.217	-0.667	0.257	0.385	0.535
25%	0.866	0.174	0.201	0.389	-0.164	0.211	1.287	0.500
35%	0.903	0.533	0.591	0.289	-0.243	0.565	2.325	0.822
40%	1.114	1.400	1.256	0.0000	-0.539	1.461	2.712	0.624
	Fractures with spatially correlated asperity contacts							
15%	0.795	0.324	0.408	0.522	-0.308	0.413	1.339	0.937
25%	0.915	0.355	0.387	0.381	-0.168	0.424	2.533	0.715
35%	1.059	0.660	0.623	0.039	-0.077	0.502	6.544	0.627
40%	0.781	0.812	1.039	0.0015	-0.958	1.660	1.733	0.245

Figure V.25 exhibits a qualitative dependence of D_s on normal stress. That is, partially saturated flow tends to be more dispersive in fractures subject to intermediate normal stress and less dispersive in fractures subject to either low or high normal stress.

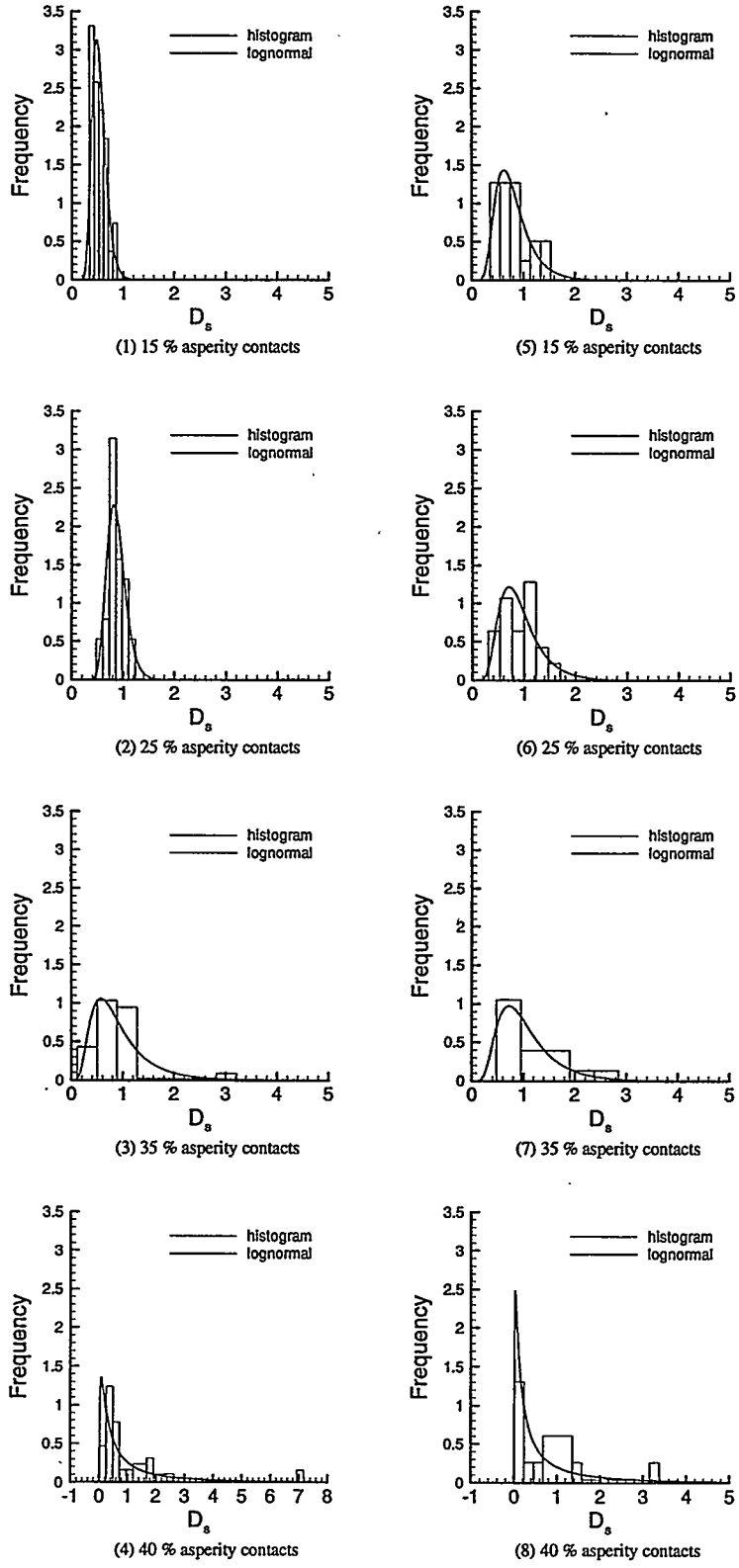


Figure V.25 Histograms of D_s for fractures with spatially random (cases 1 ~ 4) and spatially correlated (cases 5 ~ 8) asperity contacts.

This is because most fingers in fractures subject to low normal stress are not significantly affected by asperities. Thus, most realizations may have similar breakthrough behavior and the dispersion is generally weak. Similarly, only a limited number of fingers in fractures subject to high stress can break through and the dispersion is weak again. However, many fingers can be formed in fractures subject to intermediate normal stress. Some of them are fast and some of them are slow. Thus, greater value of the coefficient of seepage dispersion is generally expected in such fractures.

The seepage dispersion may be quantitatively verified by the mean value of the natural logarithm of D_s, $\overline{\ln(D_s)}$. Results in Table V.1 indicate that $\overline{\ln(D_s)}$ is smaller in fractures subject to low and high normal stress, but larger in fractures subject to intermediate normal stress. However, the arithmetic mean of D_s, i.e., $\overline{D_s}$, in fractures with spatially random asperity contacts and subject to high normal stress ($\overline{D_s}$ = 1.114) is larger than the value for fractures subject to intermediate normal stress ($\overline{D_s}$ = 0.866 or 0.903). This is because an outlier with a large value of D_s (7.169) shows up in the sample data of D_s, see the histogram in Figure V.25(4) and the corresponding flow simulation and the solid BTC in Figures V.16(c) and V.17, respectively. The presence of an outlier thus increases the standard deviation of a fitted distribution, see the columns labeled s_{D_s} and $s_{\ln(D_s)}$ in Table V.1. If the outlier were absent, however, the standard deviation would generally follow the same trend as $\overline{\ln(D_s)}$, i.e., the standard deviation of D_s would be larger for fractures at intermediate normal stress, but smaller for fractures at low and high normal stresses.

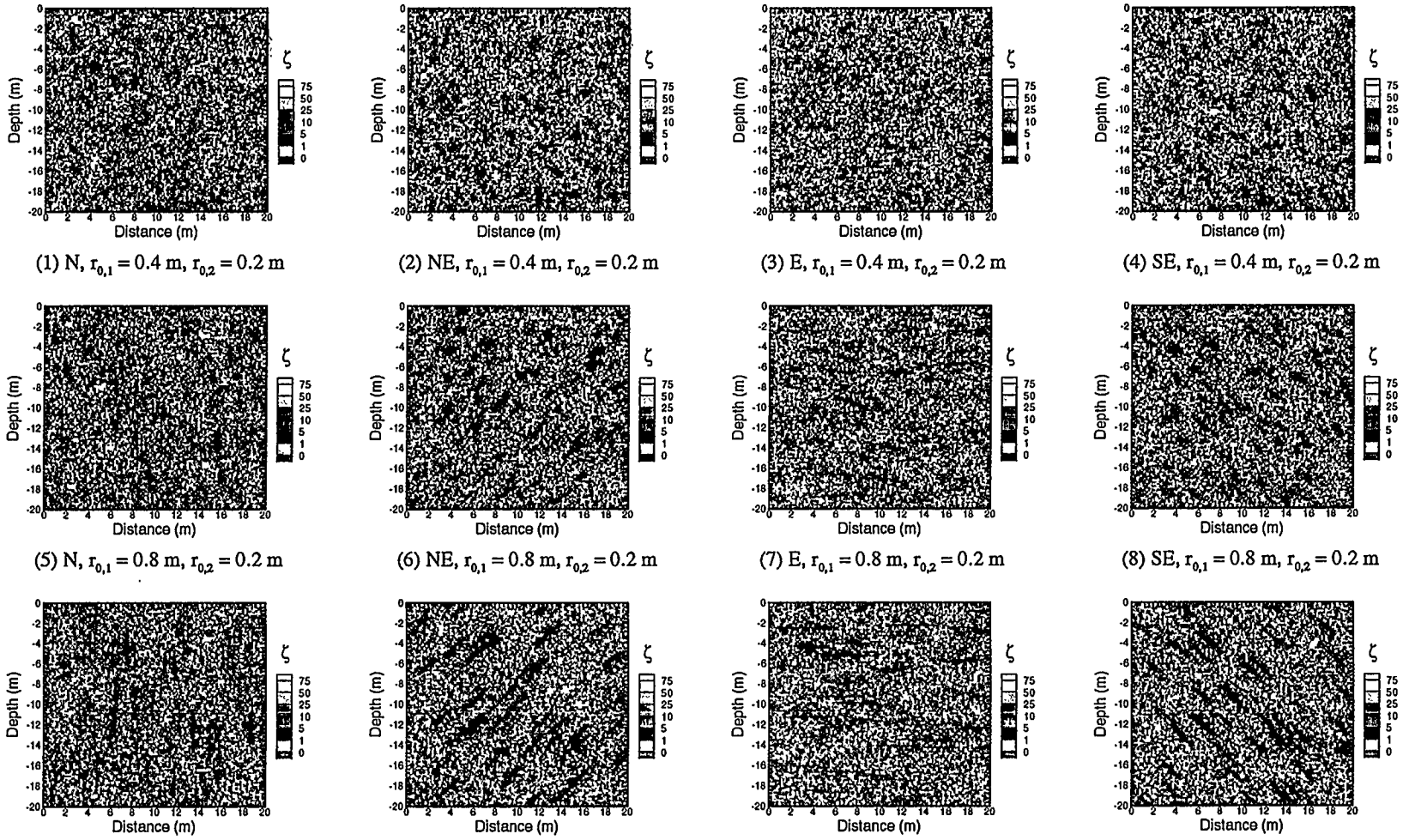
Two opposite effects of ponding on seepage dispersion may occur in fractures subject to high normal stress: (1) Ponding retards seepage as would be expected. Seepage is slowed down by ponding because liquid must fill the dead-end pores above asperities as a result of the constant recharge at the surface. This process can take a long time to complete if the total volume of dead-end pores is large. For example, flow simulations in Figures V.19(a) and V.19(b) illustrate this effect. Figure V.19(a) shows two fingers at the time of breakthrough, one to the left arrives at the bottom boundary while the other to the right reaches a depth of – 10 m. The left finger continuously evolves after breakthrough; meanwhile, the right finger proceeds downward, hits asperities and develops significant ponding, see Figure V.19(b). The gradual development of the right finger is shown on the solid BTC in Figure V.20 by a long horizontal segment, resulting in greater value of seepage dispersion, D_s = 7.169; (2) On the other hand, ponding can gather distributed seepage and funnel it into narrow paths with large fluxes and velocities. The effect of accelerated breakthrough induced by ponding can be seen on a BTC by a vertical line. Thus, weak seepage dispersion would be observed under such conditions. An example of accelerated seepage by ponding can be seen by the dashed BTC in Figure V.20 for which there is nearly no dispersion at all, i.e., D_s = 0.008. The corresponding distribution of saturation for accelerated seepage is shown in Figures V.19(c) and V.19(d). These simulation results show that all other possible flow paths are blocked by asperities and fluid is only allowed to go through one finger that finally reaches the bottom boundary.

V.5 Influence of anisotropy of asperity contacts

The purpose of these analyses is to examine the effect of anisotropy of asperity contacts on flow patterns. Anisotropic asperity contacts, whether spatially random or spatially correlated, are considered in isotropic permeability fields. Three principal correlation lengths for spatially correlated asperity contacts (or principal radii for spatially random asperity contacts), 0.4m, 0.8m, and 1.6m, are taken into account, along with the same minor correlation length (or minor radius) of 0.2m. Four principal directions, North, North-East (45° to North), East (90° to North), and South-East (135° to North), are investigated. Initial and boundary conditions are the same as those in section V.1. The total fraction of asperity contacts for these synthetic fractures is fixed at 25%. Simulated permeability fields are shown in Figures V.26 and V.27, and corresponding saturation at breakthrough and steady state are shown in Figures V.28 to V.31.

For notational convenience, the term "anisotropy ratio" in this section refers to the ratio of the principal correlation length (or radius) of anisotropic, spatially correlated (random) asperity contacts to the minor correlation length (radius) of asperity contacts. That is, three anisotropy ratios, 2.0, 4.0, and 8.0, are considered.

Results of flow simulation show that the principal direction of anisotropic asperity contacts determines the direction to which the flow is diverted. The larger the anisotropy ratio, the stronger the diversion effect. In addition, the total number of fingers reaching the bottom boundary at steady state generally decreases with increasing anisotropy ratio. This is especially true in fractures with spatially correlated asperity contacts that are not principally correlated in the vertical (North) direction. Moreover, the diversion effect

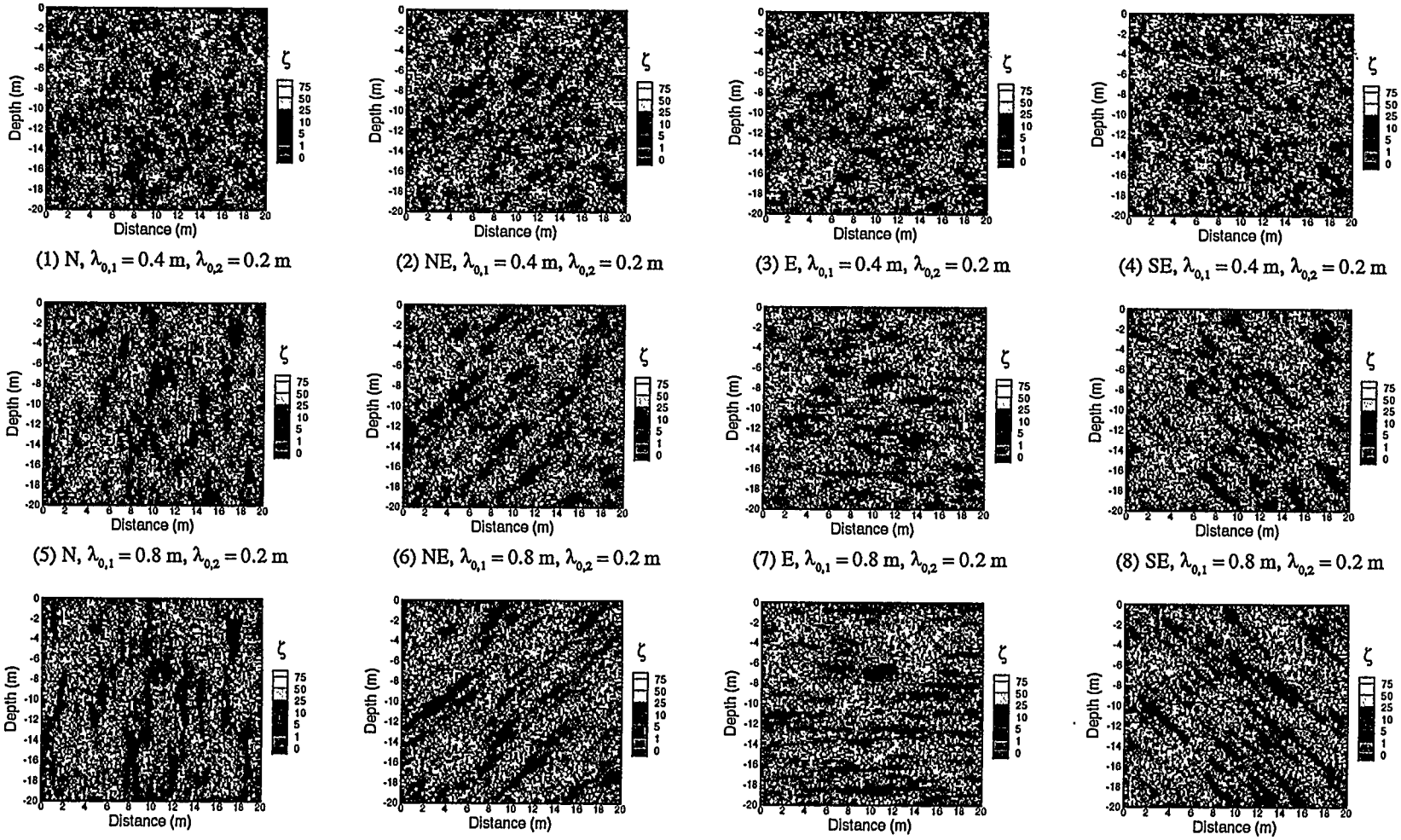


(1) N, $r_{0,1} = 0.4$ m, $r_{0,2} = 0.2$ m (2) NE, $r_{0,1} = 0.4$ m, $r_{0,2} = 0.2$ m (3) E, $r_{0,1} = 0.4$ m, $r_{0,2} = 0.2$ m (4) SE, $r_{0,1} = 0.4$ m, $r_{0,2} = 0.2$ m

(5) N, $r_{0,1} = 0.8$ m, $r_{0,2} = 0.2$ m (6) NE, $r_{0,1} = 0.8$ m, $r_{0,2} = 0.2$ m (7) E, $r_{0,1} = 0.8$ m, $r_{0,2} = 0.2$ m (8) SE, $r_{0,1} = 0.8$ m, $r_{0,2} = 0.2$ m

(9) N, $r_{0,1} = 1.6$ m, $r_{0,2} = 0.2$ m (10) NE, $r_{0,1} = 1.6$ m, $r_{0,2} = 0.2$ m (11) E, $r_{0,1} = 1.6$ m, $r_{0,2} = 0.2$ m (12) SE, $r_{0,1} = 1.6$ m, $r_{0,2} = 0.2$ m

Figure V.26 Isotropic permeability fields with anisotropic spatially random asperity contacts. Radii of asperity contacts for each case are indicated in captions. The subscripts 1 and 2 specifies the principal and minor directions of asperity contacts. Each permeability field is annealed with an isotropic semi-variogram with nugget = 0.0, sill = 190.0, and correlation length = 0.2m.



(1) N, $\lambda_{0,1} = 0.4$ m, $\lambda_{0,2} = 0.2$ m (2) NE, $\lambda_{0,1} = 0.4$ m, $\lambda_{0,2} = 0.2$ m (3) E, $\lambda_{0,1} = 0.4$ m, $\lambda_{0,2} = 0.2$ m (4) SE, $\lambda_{0,1} = 0.4$ m, $\lambda_{0,2} = 0.2$ m

(5) N, $\lambda_{0,1} = 0.8$ m, $\lambda_{0,2} = 0.2$ m (6) NE, $\lambda_{0,1} = 0.8$ m, $\lambda_{0,2} = 0.2$ m (7) E, $\lambda_{0,1} = 0.8$ m, $\lambda_{0,2} = 0.2$ m (8) SE, $\lambda_{0,1} = 0.8$ m, $\lambda_{0,2} = 0.2$ m

(9) N, $\lambda_{0,1} = 1.6$ m, $\lambda_{0,2} = 0.2$ m (10) NE, $\lambda_{0,1} = 1.6$ m, $\lambda_{0,2} = 0.2$ m (11) E, $\lambda_{0,1} = 1.6$ m, $\lambda_{0,2} = 0.2$ m (12) SE, $\lambda_{0,1} = 1.6$ m, $\lambda_{0,2} = 0.2$ m

Figure V.27 Isotropic permeability fields with anisotropic spatially correlated asperity contacts. Correlation length of asperity contacts for each case are indicated in captions. The subscripts 1 and 2 specifies the principal and minor directions of asperity contacts. Each permeability field is annealed with an isotropic semi-variogram with nugget = 0.0, sill = 120.0, and integral scale = 0.2m.

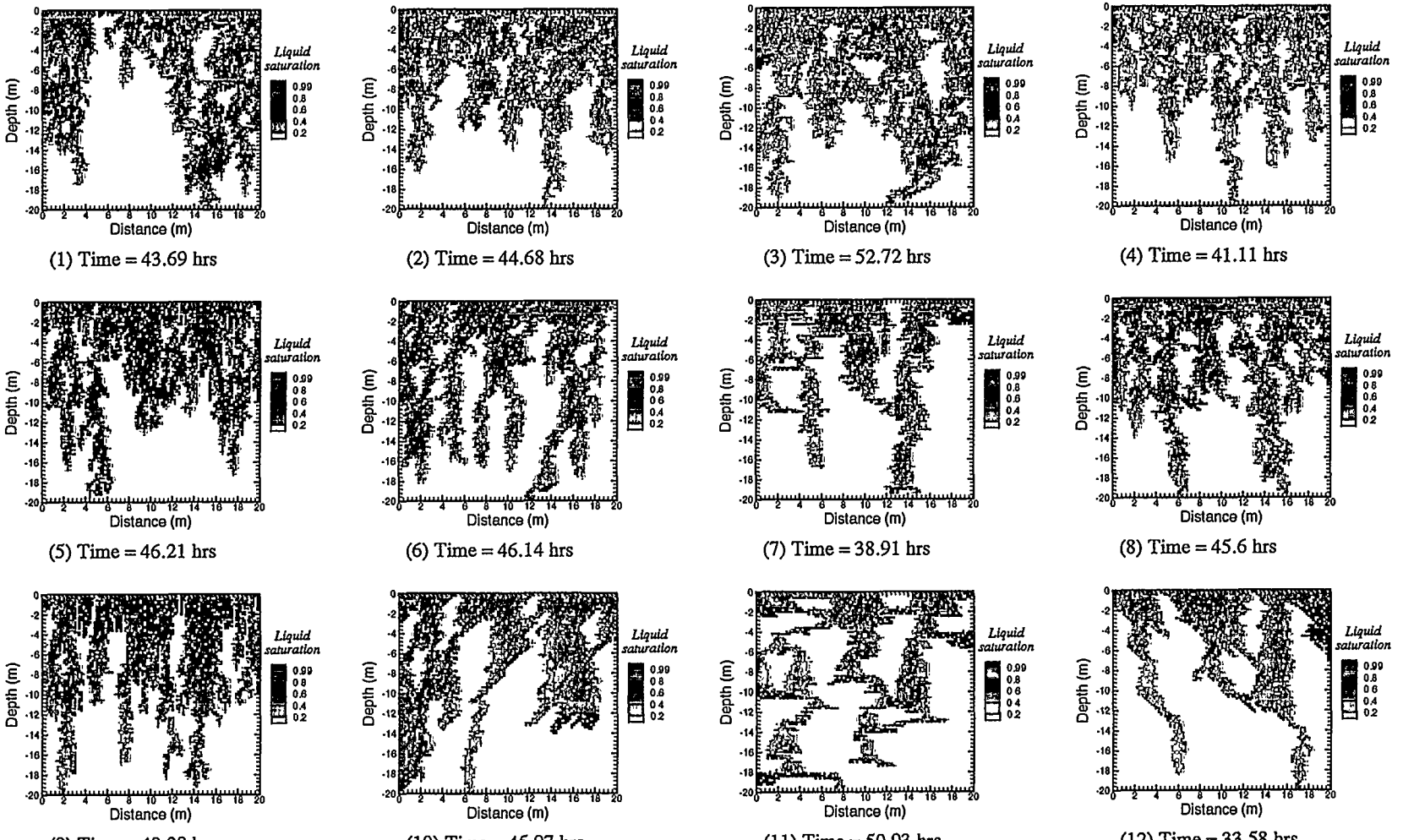


Figure V.28 Saturation at breakthrough in synthetic fractures with spatially random, anisotropic asperity contacts.

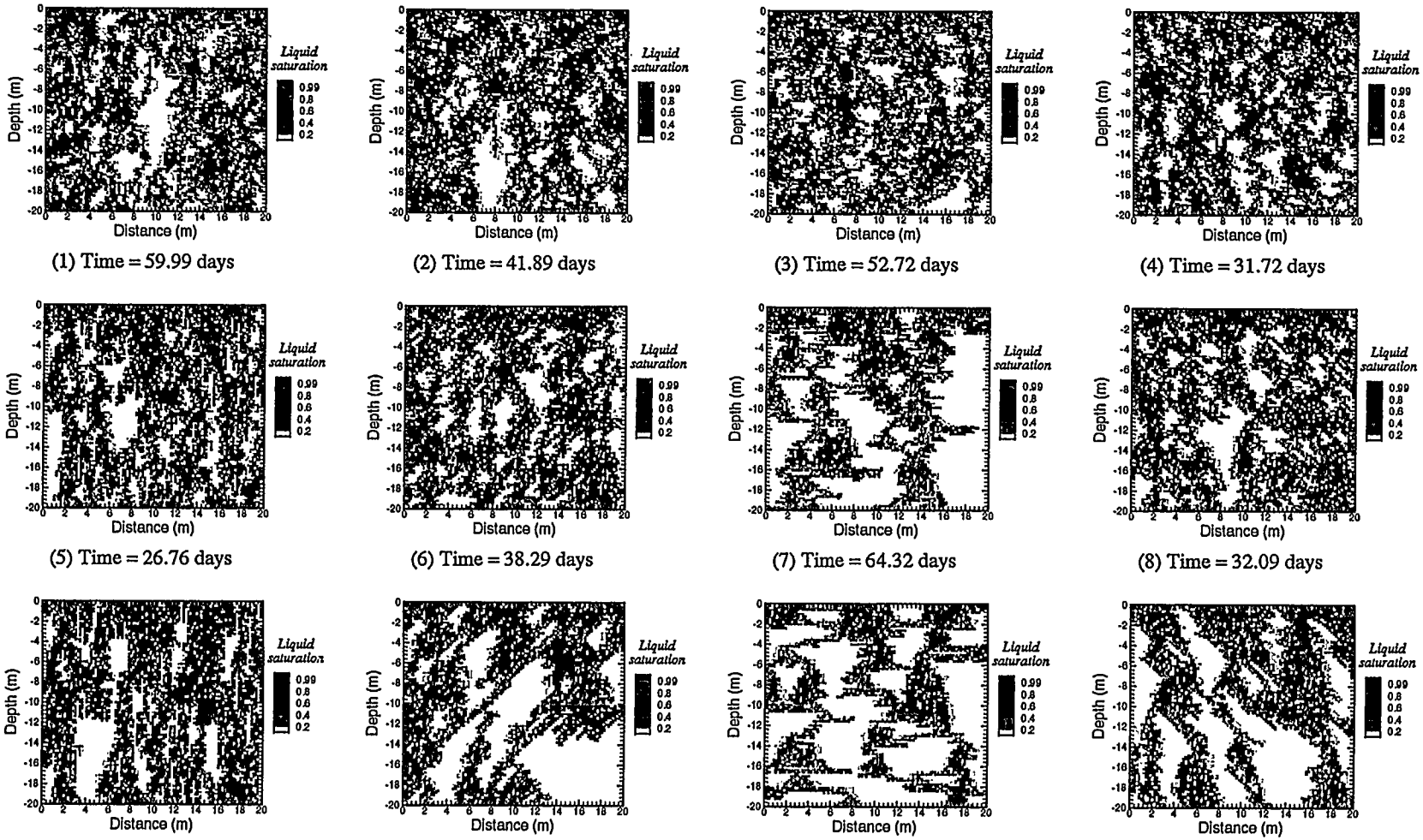


Figure V.29 Saturation at steady state in synthetic fractures with spatially random, anisotropic asperity contacts.

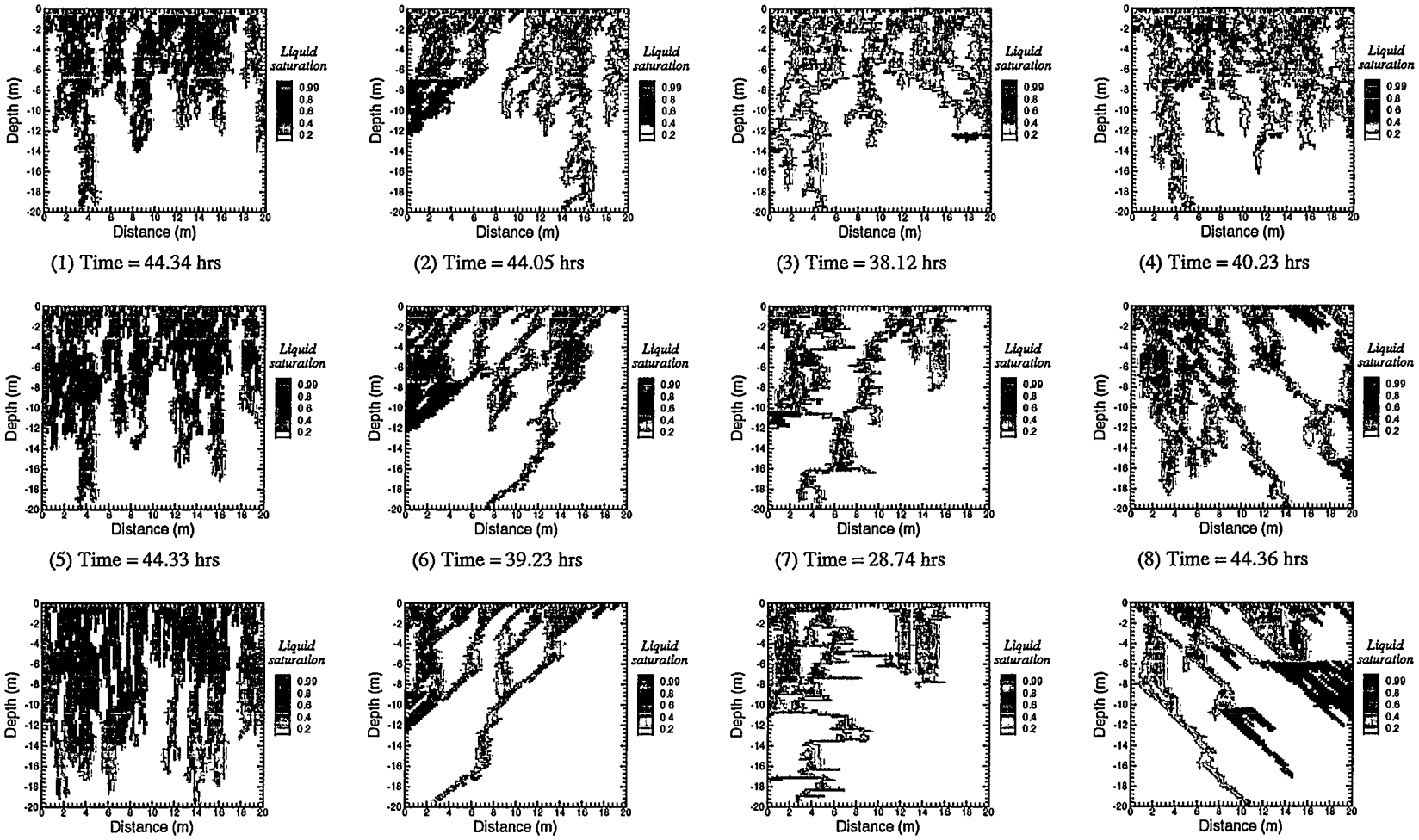


Figure V.30 Saturation at breakthrough in synthetic fractures with spatially correlated, anisotropic asperity contacts.

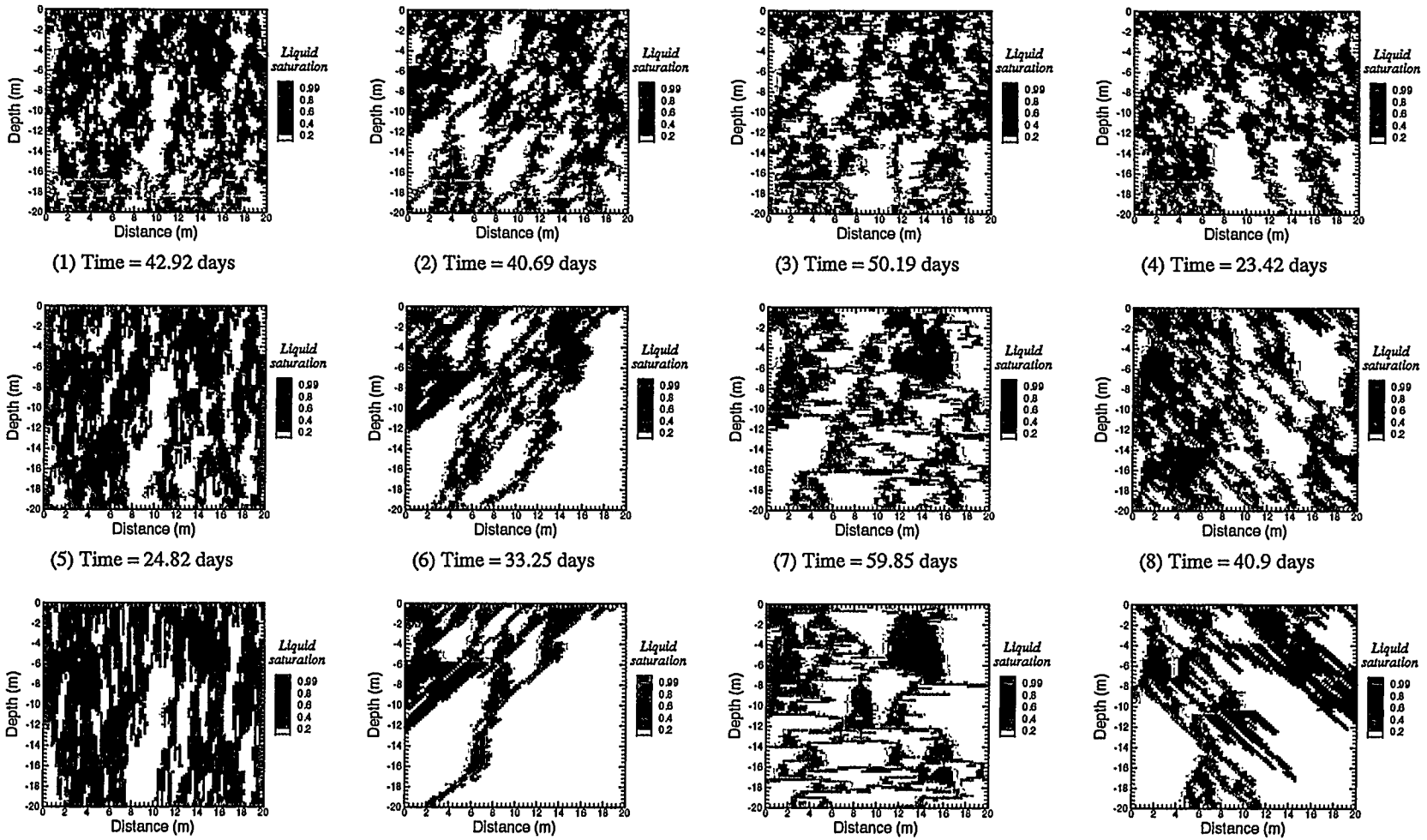
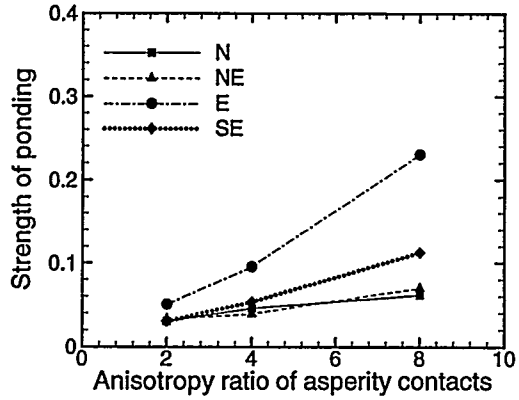


Figure V.31 Saturation at steady state in synthetic fractures with spatially correlated, anisotropic asperity contacts.

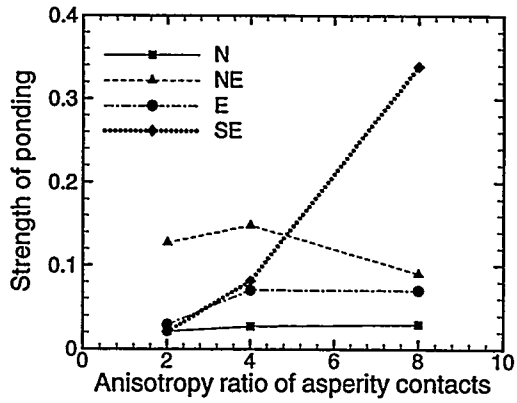
seems to be more significant in fractures with spatially correlated asperity contacts than in fractures with spatially random asperity contacts. Also, ponding effect generally increases as the anisotropy ratio increases.

The effect of ponding on seepage breakthrough is indicated in flow simulations illustrated in Figures V.30(4), V.30(8), and V.30(12). Although the vertical effective permeability in the South-East direction increases as the anisotropy ratio increases from 2.0 to 4.0, the distribution of saturation in Figure V.30(8) shows a slower breakthrough than that in Figure V.30(4). This is affected by the delaying effect of ponding, which is shown in Figure V.30(8) at the upper right and lower right corners. As the anisotropy ratio of asperity contacts increases from 4.0 to 8.0, Figures V.30(8) and V.30(12) show that more ponding is developed. In addition, calculations show that the effective permeability decreases significantly as the anisotropy ratio increases from 4.0 to 8.0. Thus, the delaying effect of ponding associated with the decreasing vertical effective permeability leads to the much slower breakthrough in Figure V.30(12).

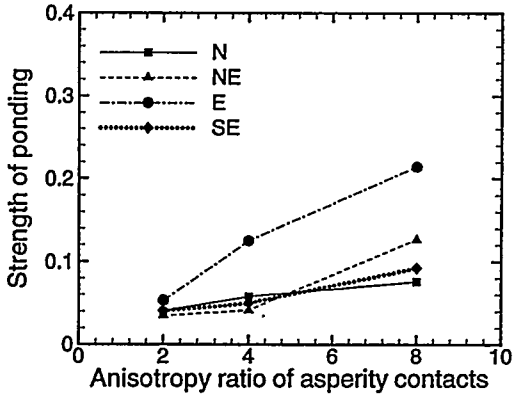
The degree of ponding as a function of anisotropy ratio is shown in Figure V.32. Generally, degree of ponding increases as the anisotropy ratio increases. In addition, degree of ponding in fractures with vertically correlated asperity contacts is generally the weakest, which becomes obvious as the anisotropy ratio increases. However, degree of ponding in fractures with non-vertically correlated asperity contacts has different levels of significance, depending on the type of asperity contacts, the principal direction, and the anisotropy ratio of asperity contacts. For example, Figures V.32(a) and V.32(b) show that the degree of ponding for horizontally correlated asperity contacts is always the



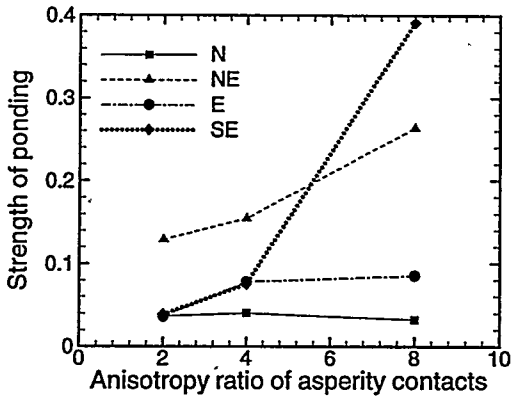
(a) Transient flow fields with spatially random asperity contacts



(c) Transients flow fields with spatially correlated asperity contacts



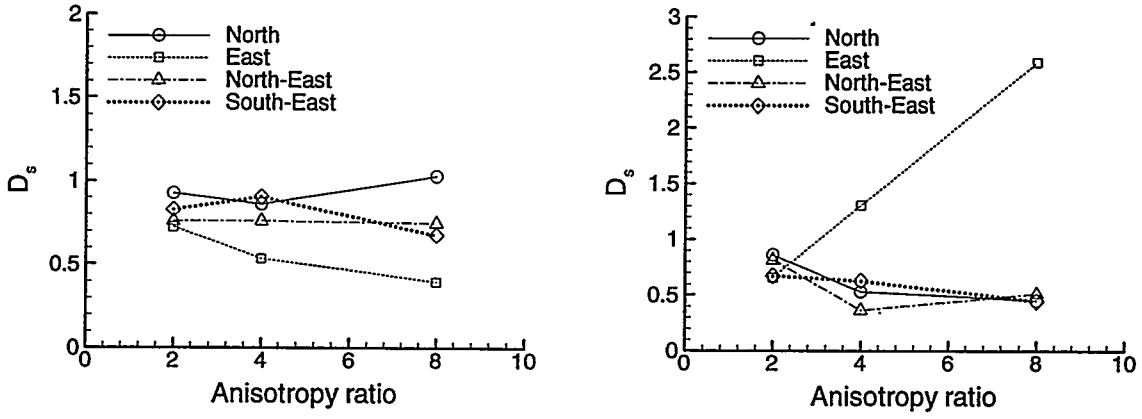
(b) Steady state flow fields with spatially random asperity contacts



(d) Steady state flow fields with spatially correlated asperity contacts

Figure V.32 Degree of ponding as a function of anisotropy ratio of asperity contacts, for transient and steady state flow fields.

greatest. However, degree of ponding strength in fractures with spatially correlated asperity contacts generally varies with the heterogeneity of permeability. Furthermore, there may be a dramatic increase in the degree of ponding in fractures with spatially correlated asperity contacts that are not vertically correlated. For example, Figure V.32(c) and V.32(d) shows that this occurs when asperity contacts are principally correlated in the South-East direction.



(a) D_s for fractures with spatially random, anisotropic asperity contacts

(b) D_s for fractures with spatially correlated, anisotropic asperity contacts

Figure V.33 Coefficient of seepage dispersion (D_s) in fractures with anisotropic asperity contacts that are spatially random or spatially correlated.

Seepage dispersion depends on the heterogeneity of fractures as well as ponding. The term "heterogeneity" herein means the overall effect of tortuosity, flow diversion, and the ability of generating new fingers after breakthrough. The later varies in different synthetic fractures but generally decreases as the anisotropy ratio increases. The general decreasing trend of seepage dispersion with increasing anisotropy ratio is observed in Figure V.33. However, the curve in Figure V.33(b) shows that seepage dispersion in the North-East direction increases as the anisotropy ratio changes from 4.0 to 8.0. This

increase is clearly due to the ponding as shown in Figures V.31(6) and V.31(10). In addition, Figure V.33(b) shows that the seepage dispersion in the horizontal direction significantly increases with the anisotropy ratio, which is the result of increasing diversion effect as illustrated in Figures V.31(3), V.31(7) and V.31(11). Similar increase of seepage dispersion with respect to anisotropy ratio also presents in Figure V.33(a) for the curve associated with the vertical correlation direction.

V. 6 Influence of anisotropy of permeability

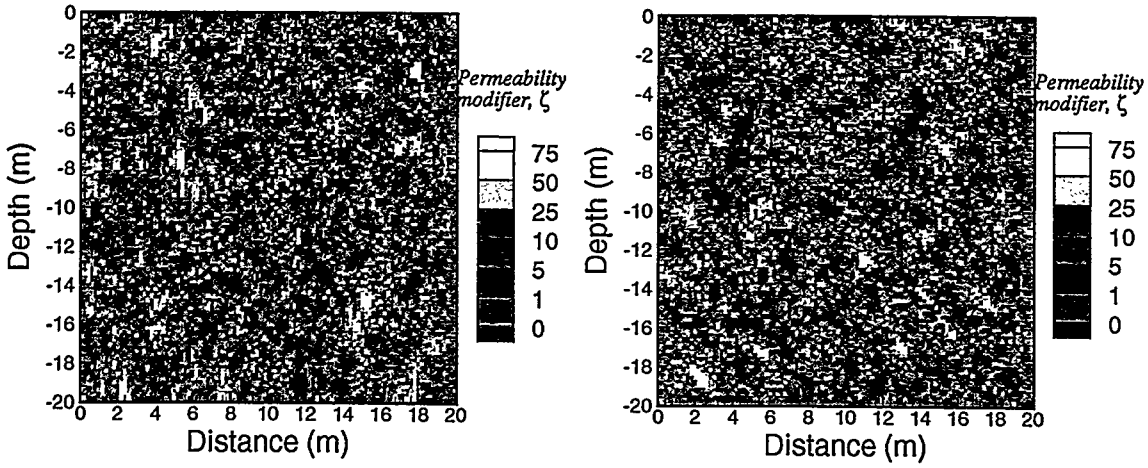
Anisotropy of the spatial distribution of asperity contacts logically leads to the consideration of anisotropy of permeability. It is then expected that the realized permeability field may have similar effects as the anisotropy of asperity contacts as far as flow bypassing and fingering are concerned.

To evaluate the effect of anisotropy of permeability, permeability fields in this section were obtained by conditioning on the same conditioning asperity contacts that were used for cases (1), (4), (9) and (12) in Figures V.26 and V.27. These anisotropic asperity contacts are principally correlated in the North or South-East directions, with principal correlation length (or principal radius) of 0.4 m or 1.6 m. Thus, anisotropic permeability with two principal correlation lengths (0.4 m and 1.6 m) and two principal directions (North and South-East) are investigated. Correlation length in the minor directions is half of its principal correlation length. Again, the total fraction of asperity contacts is fixed at 25%. Figures V.34 and V.35 are the realized permeability fields conditioned on spatially random and spatially correlated asperity contacts, respectively.

Corresponding flow simulations are shown in Figures V.36 to V.39, and the strength of ponding for these flow simulations is shown in Figure V.40.

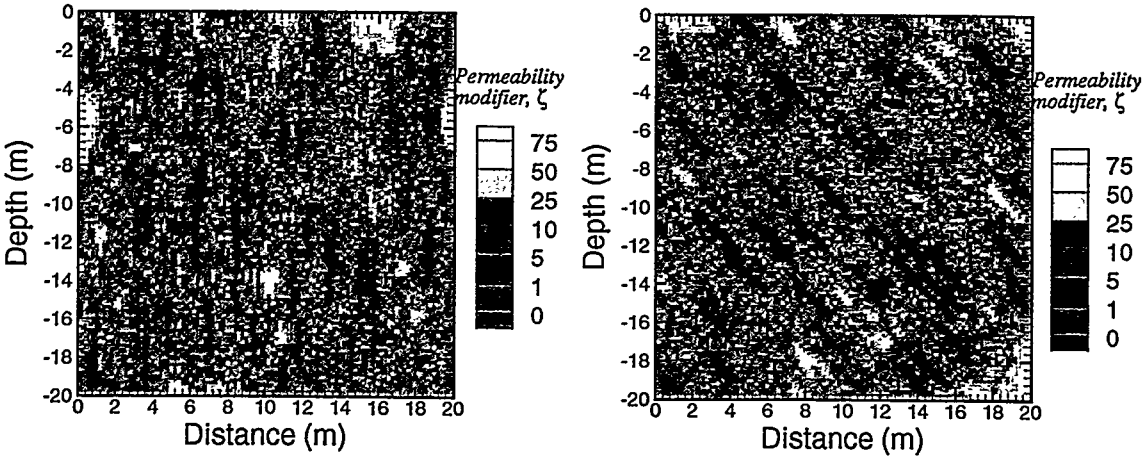
Simulation results in Figure V.38 (or Figure V.39) closely resemble the simulation results (1), (4), (9) and (12) in Figure V.30 (or Figure V.31). Recall that the former and the later simulation scenarios were carried out in synthetic fractures conditioned on the same spatially correlated asperity contacts but different correlation structures of permeability. This similarity thus suggests that liquid seepage in natural fractures may be less sensitive to the spatial correlation of permeability and tends to depend more strongly on the spatial distribution of asperity contacts. It also suggests that tortuosity of flow paths may be insensitive to the anisotropy of permeability as long as conditioning asperity contacts are the same. However, the difference between the two flow scenarios can be identified in terms of degree of ponding as illustrated in Figure V.40. It shows that the two flow scenarios are obviously different from each other if the asperity contacts are not vertically correlated, see Figures V.40(e) to V.40(h).

Breakthrough and steady state flow fields shown in Figures V.36 and V.37 are also similar to corresponding simulation results, i.e., cases 1, 4, 9, and 12 in Figures V.28 and V.29. The minor difference between these simulation results is because some of the spatially random asperity contacts in these fractures are not conditioning data. That is, the overall spatial distribution of asperity contacts is not the same between the two flow scenarios. The difference becomes significant only for cases with larger anisotropy ratios. For example, the ponding strength in Figure V.40(b) and V.40(d) explicitly show the difference between the two simulation scenarios.



(a) $r_0 = 0.4$ m, $\lambda_{k,1} = 0.4$ m, $\lambda_{k,2} = 0.2$m
principal direction = N-S

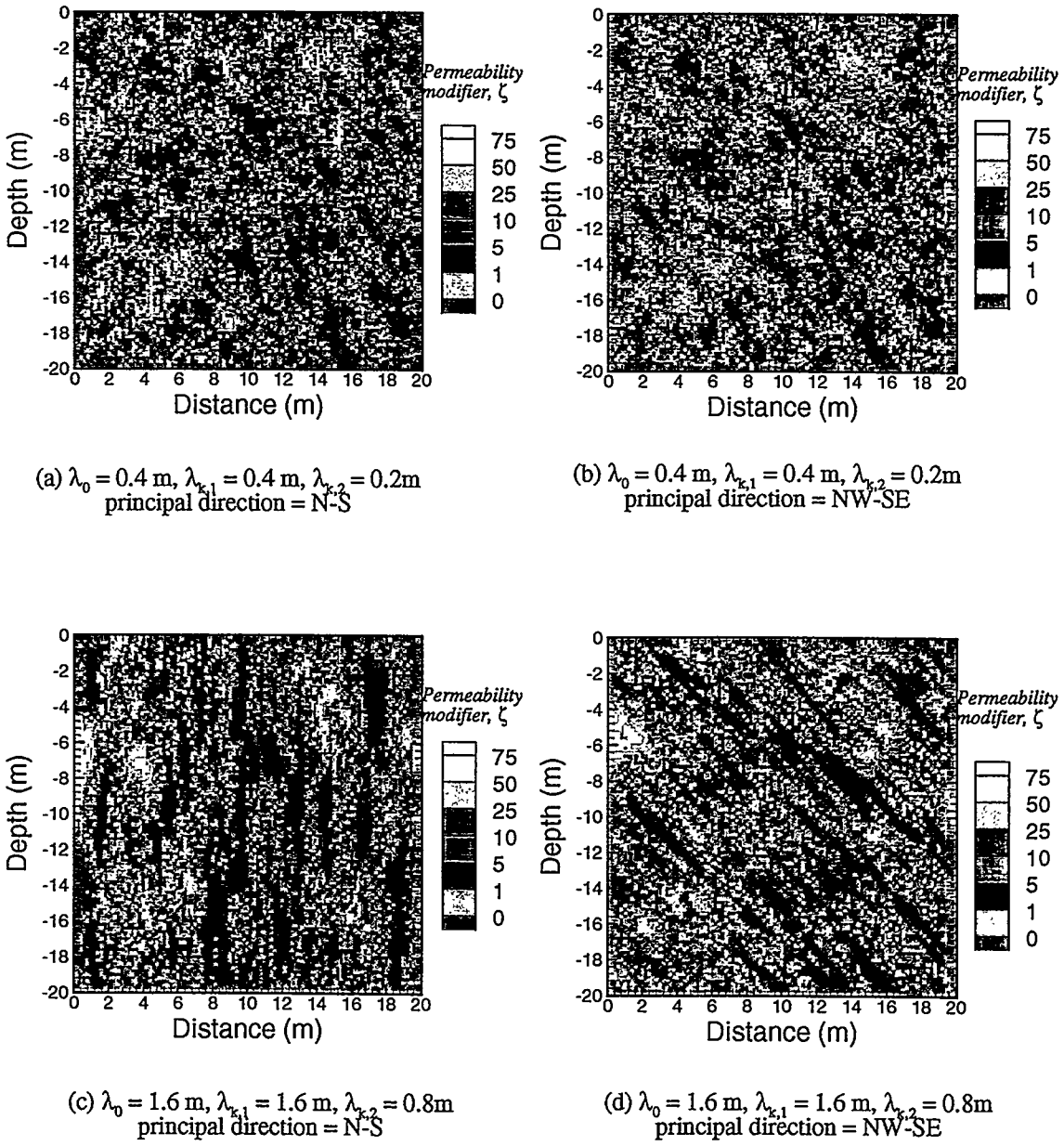
(b) $r_0 = 0.4$ m, $\lambda_{k,1} = 0.4$ m, $\lambda_{k,2} = 0.2$m
principal direction = NW-SE



(c) $\lambda_0 = 1.6$ m, $\lambda_{k,1} = 1.6$ m, $\lambda_{k,2} = 0.8$m
principal direction = N-S

(d) $r_0 = 1.6$ m, $\lambda_{k,1} = 1.6$ m, $\lambda_{k,2} = 0.8$m
principal direction = NW-SE

Figure V.34 Anisotropic permeability fields conditioned on spatially random asperity contacts. The principal radii of asperity contacts are 0.4m or 1.6m, and principal directions in N-S or NW-SE. The anisotropic semi-variogram of permeability has nugget = 0, sill = 190, and principal correlation length as 0.4m or 1.6m. The minor correlation length is half of the principal correlation length. The subscripts 1 and 2 for λ_k denote the principal and minor directions, respectively.



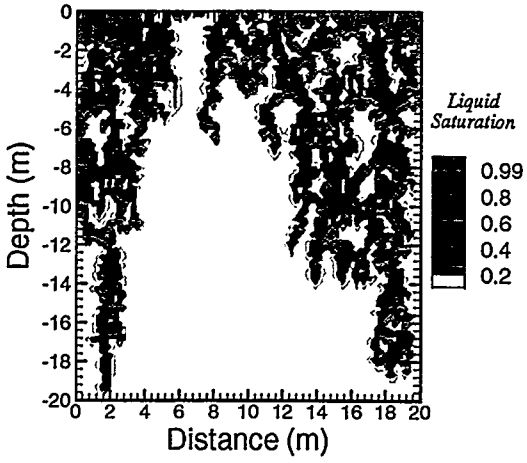
(a) $\lambda_0 = 0.4$ m, $\lambda_{k,1} = 0.4$ m, $\lambda_{k,2} = 0.2$m
principal direction = N-S

(b) $\lambda_0 = 0.4$ m, $\lambda_{k,1} = 0.4$ m, $\lambda_{k,2} = 0.2$m
principal direction = NW-SE

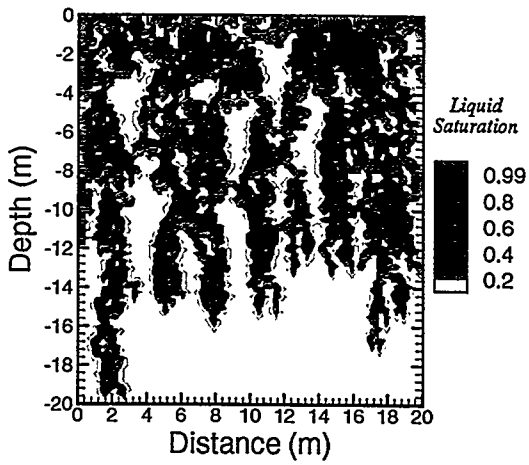
(c) $\lambda_0 = 1.6$ m, $\lambda_{k,1} = 1.6$ m, $\lambda_{k,2} = 0.8$m
principal direction = N-S

(d) $\lambda_0 = 1.6$ m, $\lambda_{k,1} = 1.6$ m, $\lambda_{k,2} = 0.8$m
principal direction = NW-SE

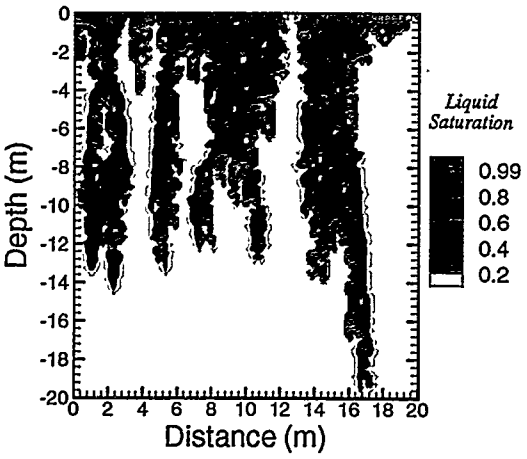
Figure V.35 Anisotropic permeability fields conditioned on spatially correlated asperity contacts. The principal correlation lengths of asperity contacts are 0.4m or 1.6m, and principal directions in N-S or NW-SE. The anisotropic semi-variogram of permeability has nugget = 0, sill = 120, and principal correlation length as 0.4m or 1.6m. The minor correlation length is half of the principal correlation length. The subscripts 1 and 2 for λ_k denote the principal and minor directions, respectively.



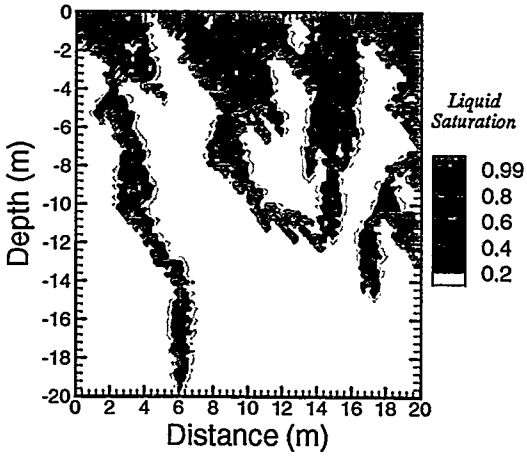
(a) Time = 39.48 hrs



(b) Time = 48.17 hrs

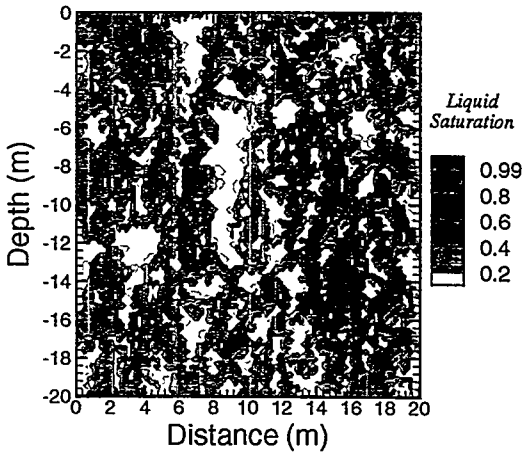


(c) Time = 38.48 hrs

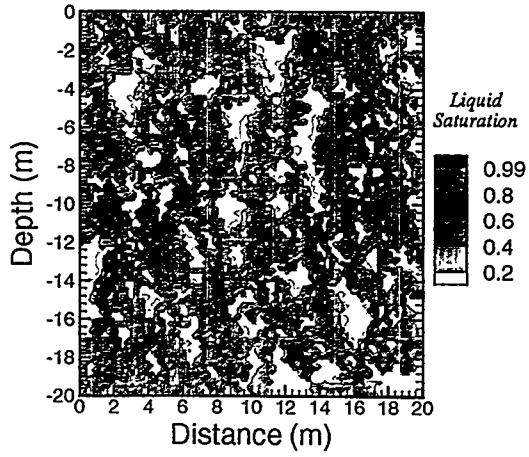


(d) Time = 34.35 hrs

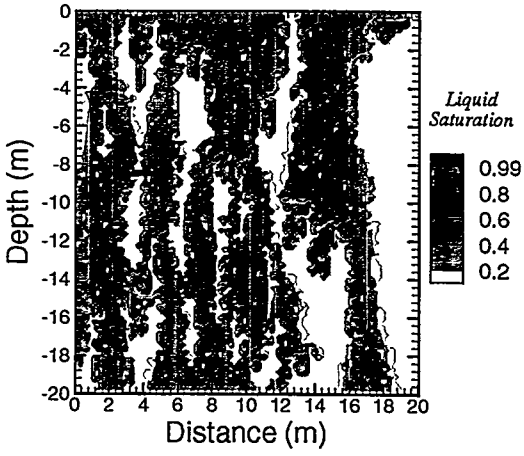
Figure V.36 Saturation at breakthrough in anisotropic fractures with spatially random, anisotropic asperity contacts shown in Figure V.34.



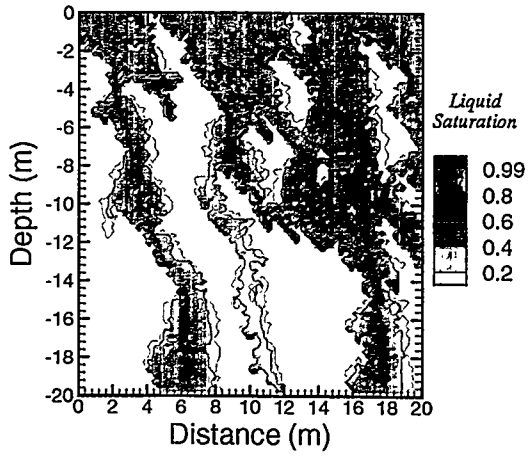
(a) Time = 56.95 days



(b) Time = 28.76 days

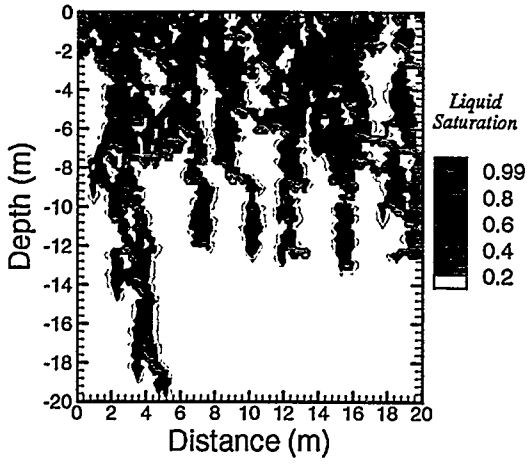


(c) Time = 28.35 days

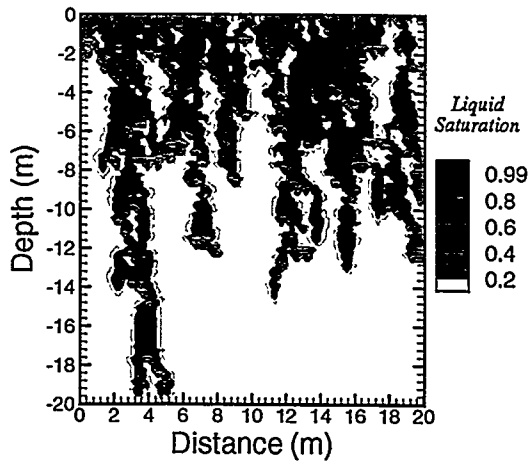


(d) Time = 72.29 days

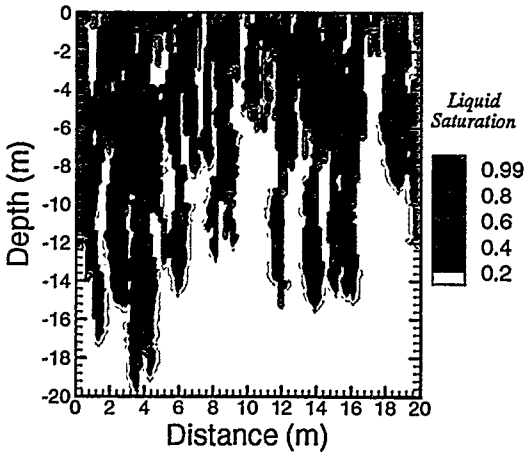
Figure V.37 Saturation at steady state in anisotropic fractures with spatially random, anisotropic asperity contacts shown in Figure V.34.



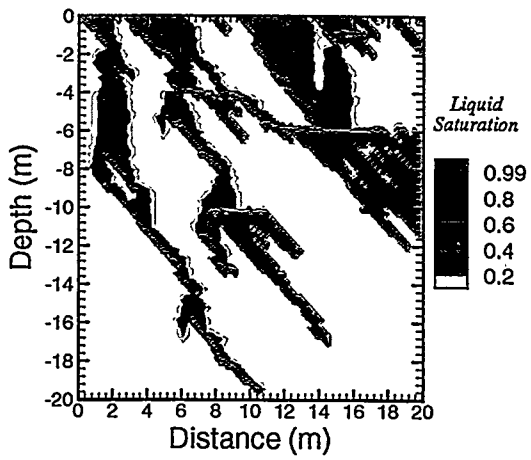
(a) Time = 38.37 hrs



(b) Time = 38.06 hrs

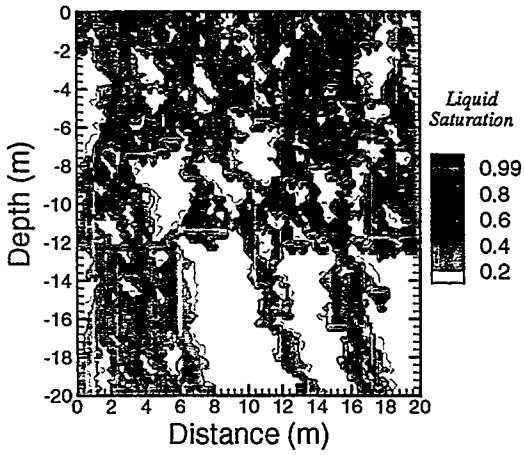


(c) Time = 44.46 hrs

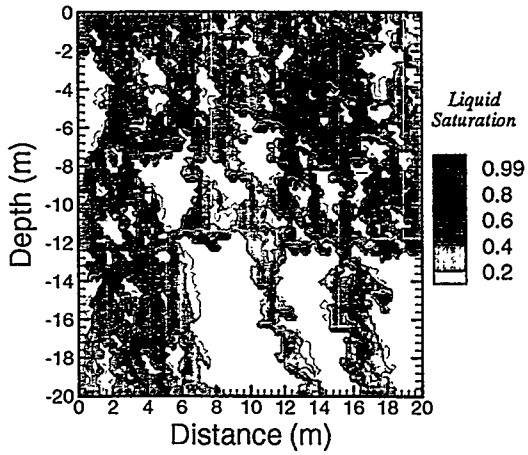


(d) Time =52.20 hrs

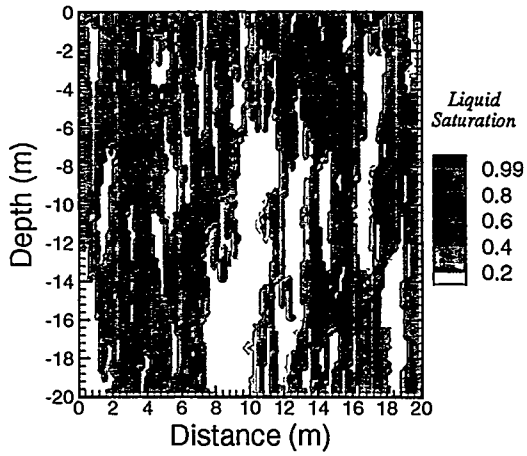
Figure V.38 Saturation at breakthrough in anisotropic fractures with spatially correlated, anisotropic asperity contacts shown in Figure V.35.



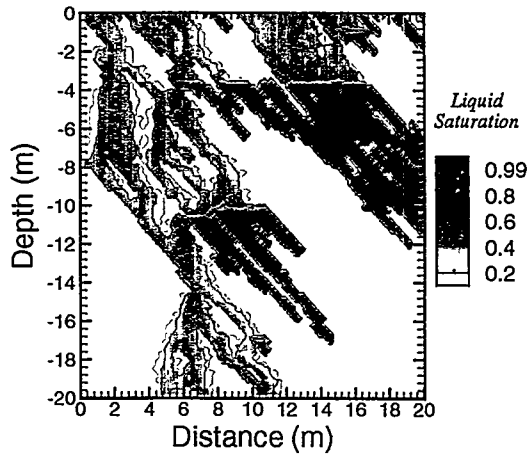
(a) Time = 28.79 days



(b) Time = 31.95 days

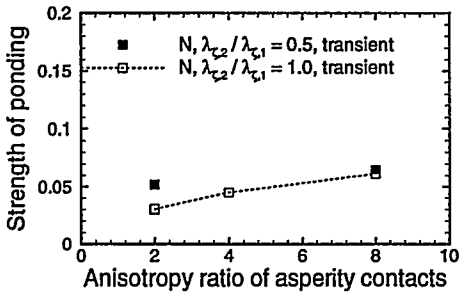


(c) Time = 19.33 days

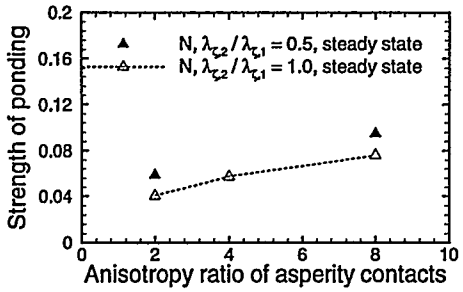


(d) Time = 23.42 days

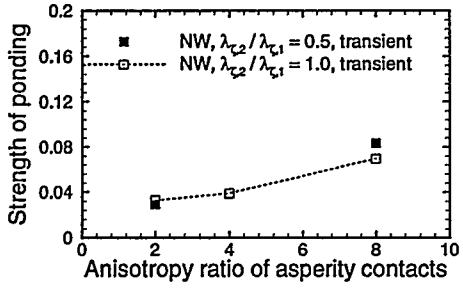
Figure V.39 Saturation at steady state in anisotropic fractures with spatially correlated, anisotropic asperity contacts shown in Figure V.35.



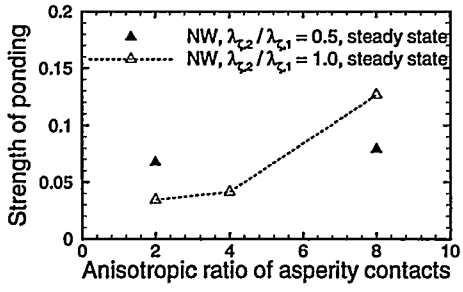
(a) N, spatially random asperity contacs



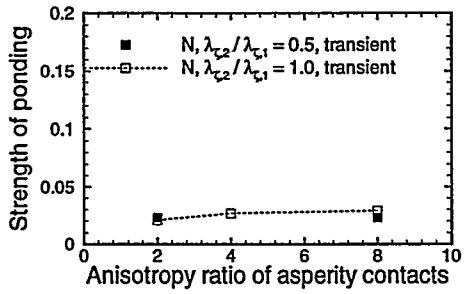
(b) N, spatially random asperity contacs



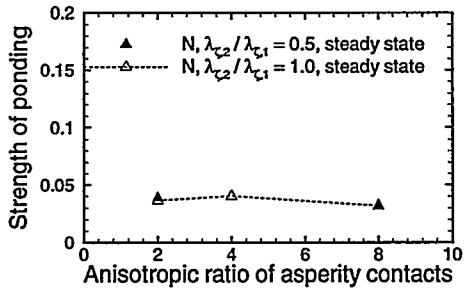
(c) NW, spatially random asperity contacs



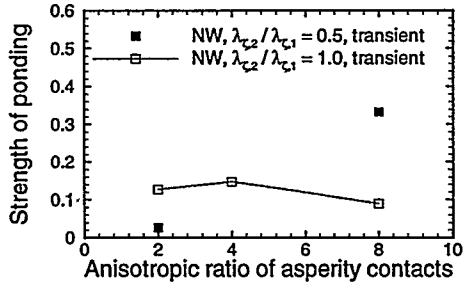
(d) NW, spatially random asperity contacs



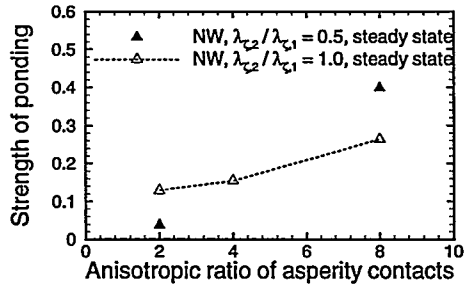
(e) N, spatially correlated asperity contacts



(f) N, spatially correlated asperity contacts



(g) NW, spatially correlated asperity contacts



(h) NW, spatially correlated asperity contacts

Figure V.40 Comparisons of the degree of ponding in two flow scenarios, one with isotropic permeability and the other with anisotropic permeability. Synthetic fractures for both flow scenarios are conditioned on the same anisotropic asperity contacts.

Chapter VI. Tracer simulations

VI. 1 Tracer transport

As stated earlier, field evidence suggests that, in semi-arid environments, water is able to migrate downward rather rapidly along preferential paths. At Yucca Mountain, Nevada, for example, fracture networks in unsaturated rocks may provide such fast preferential flow paths. This observation thus raises concerns that, once a storage canister starts leaking, radionuclides may transport with groundwater and reach the downstream biosphere at an unexpectedly short time scale. Detailed analyses of solute transport need to consider coupled flow and transport equations, which are beyond the scope of the present study. To simplify the problem, however, tracer transport without taking into account the effect of molecular diffusion is considered in this section.

"Tracers" are defined herein as dilute compounds that are completely non-reactive and dissolved in water without significantly changing its physical properties, e.g., density and viscosity, of water. An example would be a small amount of brine will mixed with water. By this convention, tracers can be treated as a second component of water.

For multi-phase simulations, liquid saturation is defined as the ratio of pore volume occupied by the liquid phase to the total pore volume, i.e.,

$$S_l = \frac{\mathrm{V}_l}{\mathrm{PV}} = \frac{\mathrm{V}_l}{\phi \mathrm{V}_\mathrm{b}} \tag{6.1}$$

where V_l is the pore volume occupied by liquid phase, PV stands for the total pore volume, ϕ is porosity, and V_b is the bulk volume of rock. Primary variables for two-

component simulations are defined as liquid saturation (S_l) (or liquid pressure (P_l)) and mass fraction of the second component of liquid phase(X_2) (Pruess, 1991). Mass fractions for two-component flow systems are defined as

$$X_i = \frac{M_i}{\sum_{i=1}^{2} M_i} = \frac{\rho_l V_{l,i}}{\sum_{i=1}^{2} \rho_l V_{l,i}} = \frac{V_{l,i}}{V_l} = \frac{V_{l,i}}{PV} \cdot \frac{1}{S_l} \tag{6.2}$$

If the i-th component of liquid saturation is defined as

$$S_{l,i} = \frac{V_{l,i}}{PV} \tag{6.3}$$

then the mass fraction and liquid saturation have the following relationship

$$S_{l,i} = X_i \, S_l \tag{6.4}$$

See Figure VI.1 for illustration of two-water systems.

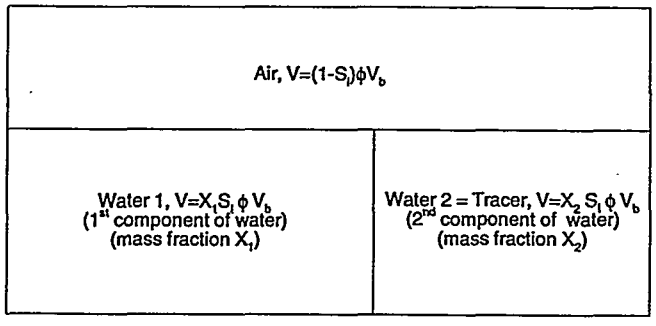


Figure VI.1 Schematic partition of the pore space for two-water systems.

For i= 2, i.e., the second component of water; or tracer, Eq(6.4) yields

$$S_{l,2} = X_2 \, S_l \tag{6.5}$$

That is, the quantity $S_{l,2}$ is in fact the pore volume occupied by tracer per unit void volume of rock. From Eq(6.5), the volumetric fraction of tracer to the total pore volume is the mass fraction of tracer, $S_{l,2}$. Strictly speaking, $S_{l,2}$ is not liquid saturation but a "pore-space-weighted saturation". For notational convenience, however, $S_{l,2}$ will be named "tracer saturation" in the remainder of the text.

Initial conditions for tracer transport may be a steady state or a transient flow field of water. First, let us consider the steady state flow field such as Figure V.19(d). This flow field is interesting because (1) it is the result of water simulation in a synthetic fracture at high normal stress, (2) it has the least seepage dispersion, and (3) it has several regions that are fully saturated with water. These fully saturated regions have different effects on seepage for transient and steady state flow fields. For example, the effects of seepage retardation and seepage acceleration by ponding are illustrated by the horizontal segment in Figure VI.2 and the dashed BTC in Figure V.20, respectively. Moreover, some of the saturated regions may become "dead-end pores" to water. Note that dead-end pores usually occur above laterally extensive asperity contacts or at fracture terminations, e.g., the upper left corner of Figure V.19(d) and to the right at depths from – 3 m to – 8 m.

Figure VI.3 shows four snapshots of tracer transport at different times. These simulations were obtained by injecting tracer (at the constant rate of 10^{-3} kg/s uniformly distributed over the entire top boundary) into the steady state flow field of Figure V.19(d) and simulating under the same boundary conditions as specified in section V.1. These

simulations were terminated at the times when the flux ratios of tracer reached 0.1, 0.5, 0.9, and 0.987, respectively. The flux ratio for tracer simulations is defined as

$$\frac{Q_{2,bot}}{Q_{top}} = \frac{\sum_j X_{2,j} Q_{bot}}{Q_{top}} \tag{6.6}$$

where Q_{top} and Q_{bot} are the total liquid flux at the top and bottom boundary, respectively. $Q_{2,bot}$ is the total tracer flux at the bottom boundary and is defined as the numerator in Eq(6.6). The subscript j in Eq(6.6) stands for all grid blocks that directly contact the bottom boundary, and X_{2j} is the mass fraction of tracer in grid block j.

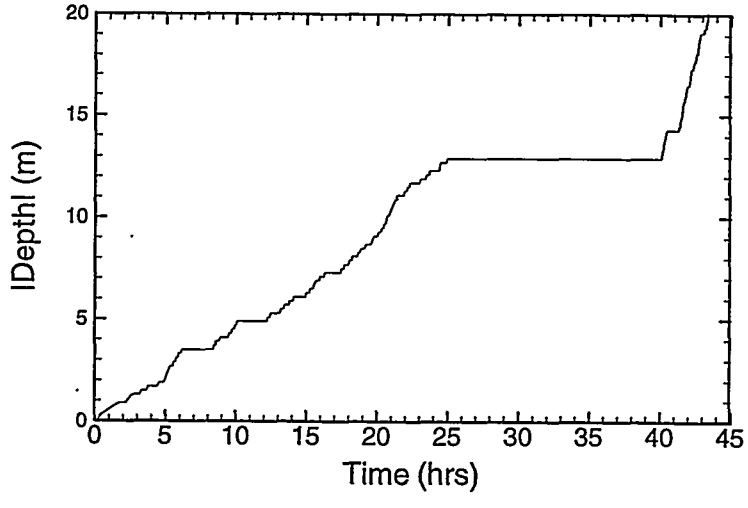
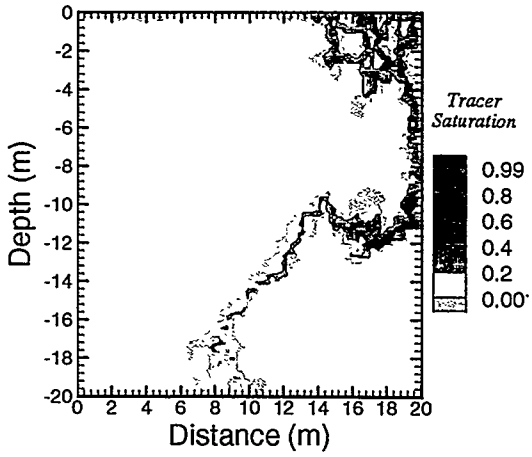
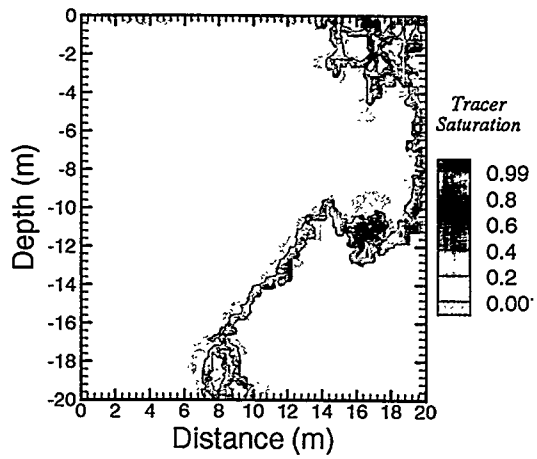


Figure VI.2 Vertical advance curve for the realization shown in Figure V.19(d).

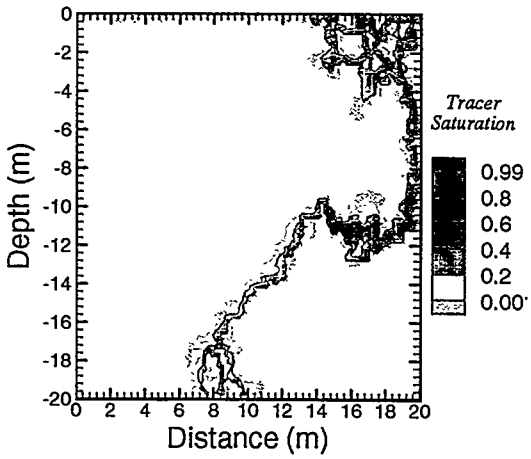
Simulation results in Figure VI.3 show that tracer not only travels along flow paths that have been developed by water but bypasses dead-end pores. Obviously, tracer flow paths have higher relative permeability (or, equivalently, lower flow resistance) due to higher total liquid saturation. The bypassing of tracer away from ponded regions is because of the assumption of non-reactive tracers and the neglect of molecular diffusion.



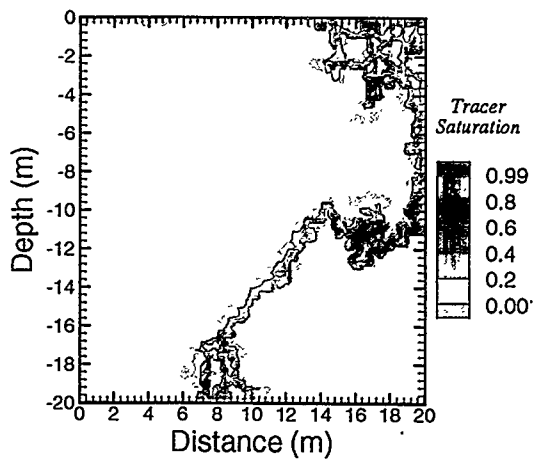
(a) $Q_{2,bot}/Q_{top} = 0.1$, Time = 0.63 days



(c) $Q_{2,bot}/Q_{top} = 0.9$, Time = 1.68 days



(b) $Q_{2,bot}/Q_{top} = 0.5$, Time = 0.91 days



(d) $Q_{2,bot}/Q_{top} = 0.987$, Time = 4.62 days

Figure VI.3 Snapshots of tracer transport with the steady state flow field shown in Figure V.19(d) as the initial condition.

Neglecting molecular diffusion thus limits the time scale for which the realistic solute transport can occur. The limiting time scale can be estimated by the diffusive length scale $y = \sqrt{2Dt}$, where D is a typical aqueous phase diffusivity of order 10^{-10} m^2/s. For example, the time scale for tracers to diffuse into the ponded region in Figure VI.3(d) from depths – 11 m to – 10 m would approximately take 160 years. Since the simulation time for Figure VI.3(d) is only 4.56 days, which is far below the limiting time scale, tracers do not invade those dead-end pores by diffusion. Thus, the invasion of tracer into the ponded region above the depth at –12 m is due to other mixing mechanisms. In this study, the only mixing mechanism occurring between water and tracer is caused by the finite spatial resolution in the computational grids, with a dispersivity on the order of $\Delta z/2$ = 0.1 m (Pruess, 1991). The quantitative measurement of the mixing is then represented by mass fractions.

Breakthrough curves of water and tracer are shown in Figure VI.4. As discussed earlier in section V.3 the BTC for liquid seep in Figure VI.19(d) is delayed by ponding, and it behaves likes a step function. However, the tracer BTC is smoother than the water BTC. Moreover, the tracer BTC is more dispersive than the water BTC and shows the effects of dispersion.

Figure VI.5 shows the results of fitting the log-normal travel time model to each of the BTCs in Figure VI.4. The sample BTC as obtained from flow simulations is plotted as solid dots in Figure VI.5. Recall that analytical pdf and CDF are obtained by calculating the sample mean and sample variance from the sample BTC data, see section V.3, which are plotted as solid and dashed-dotted lines respectively in Figure VI.5. As

expected, the log-normal model does not fit the water BTC very well, see Figure VI.5(a). Figure VI.5(b) shows that the log-normal model fits the tracer BTC very well, especially for the analytical CDF. In addition, the probability distribution for tracer travel time tends to be a positively skewed distribution. This property of tracer BTC suggests that the time scale for the total breakthrough is longer for tracer transport than for water transport.

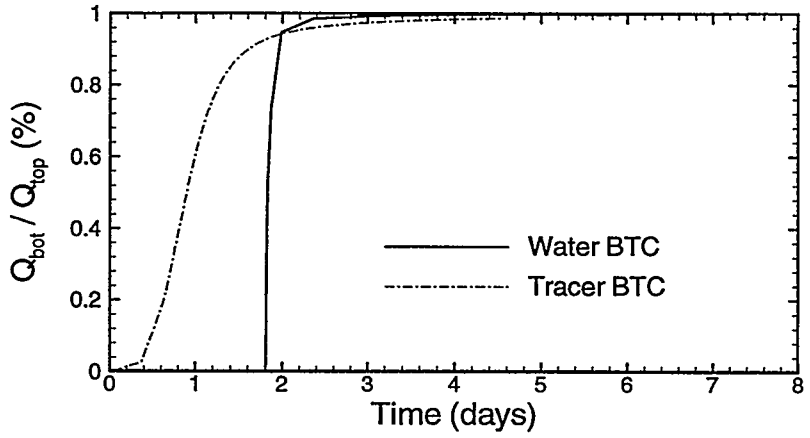
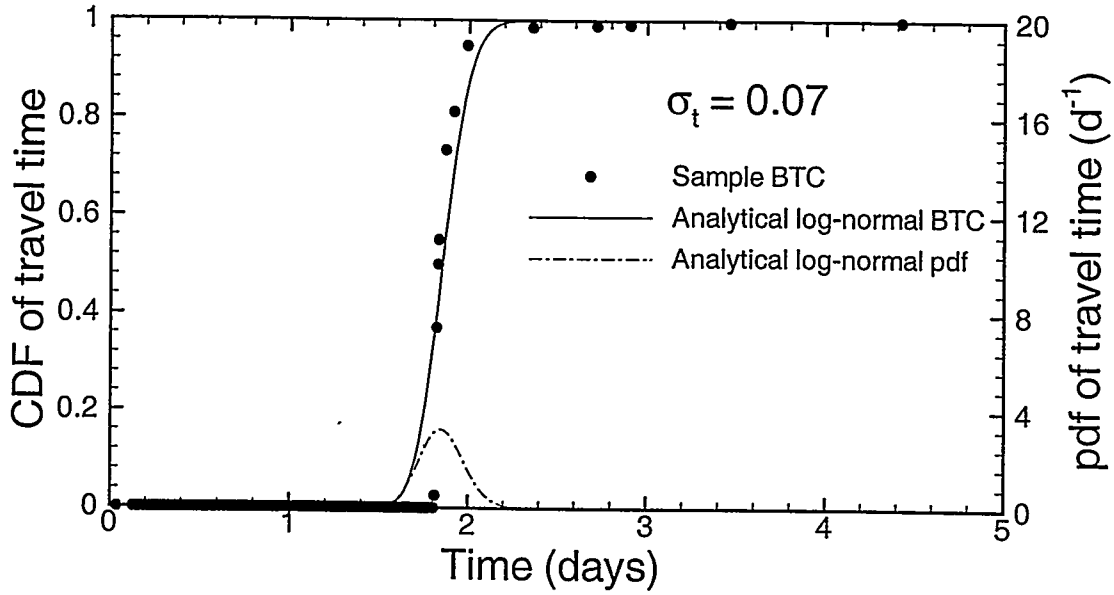
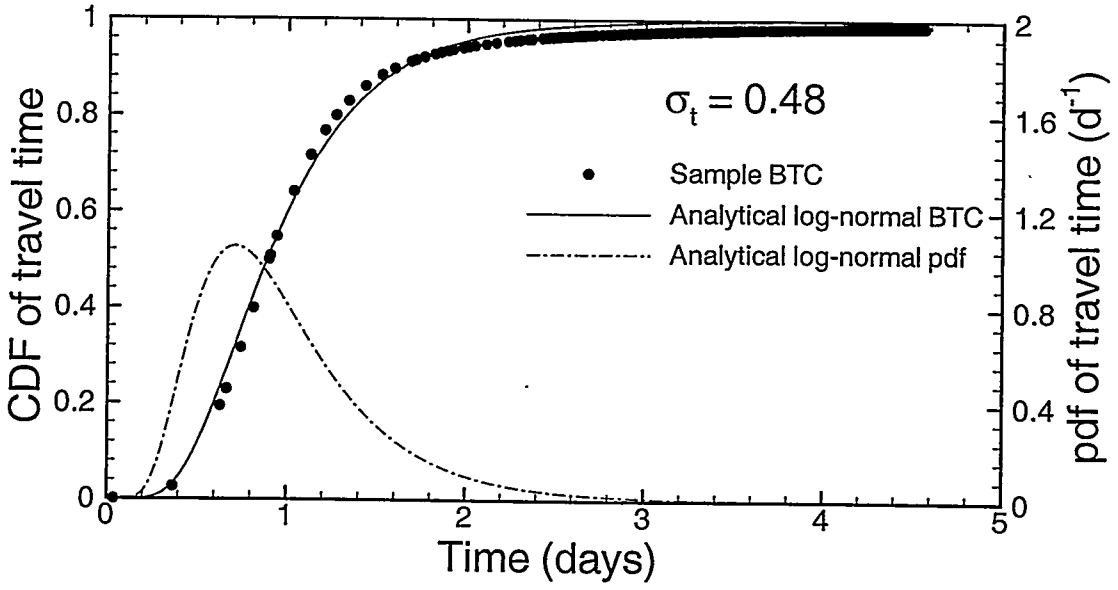


Figure VI.4 BTCs of water and tracer transport shown in Figures V.19(d) and VI.3(d).

The next example investigates the effect of initial condition on tracer transport. Instead of starting from a steady state flow field of water, this numerical experiment uses a transient flow field as the initial condition. Figure VI.6(a) shows the initial condition for this experiment. This flow field was obtained by injecting water uniformly over the entire top boundary into a synthetic fracture with spatially correlated asperity contacts. The constant injection rate of water was 10^{-3} kg/s, and the total fraction of asperity contacts was 40%. In addition, this transient flow field was terminated at the time at which the flux ratio is approximately 0.5. For comparison, its corresponding steady state flow field is shown in Figure VI.6(b).

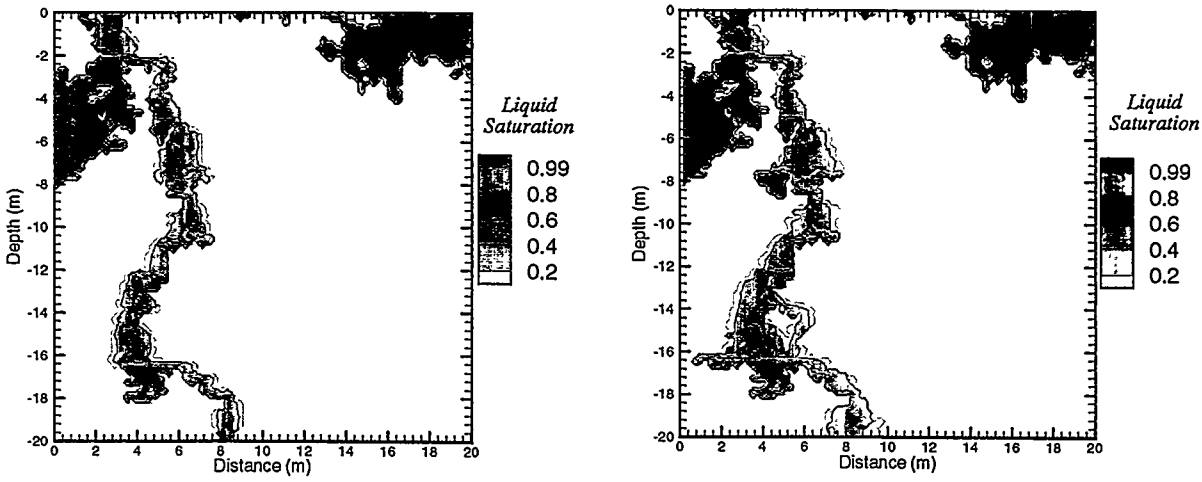


(a) Water BTC



(b) Tracer BTC

Figure VI.5 Fitted results of the log-normal travel time model to the water and tracer BTCs shown in Figure VI.4.



(a) Transient flow field of water at 0.42 days after breakthrough

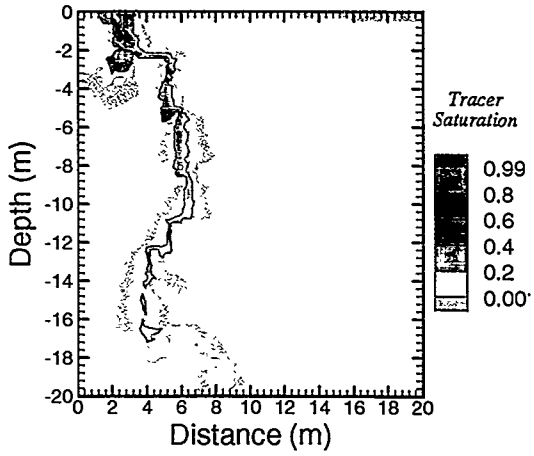
(b) steady state flow field of water at 12.85 days after injection

Figure VI.6 Initial conditions for tracer simulations: (a) the transient flow field of water, which is approximately terminated at flux ratio = 0.5, and (b) its corresponding steady state flow field.

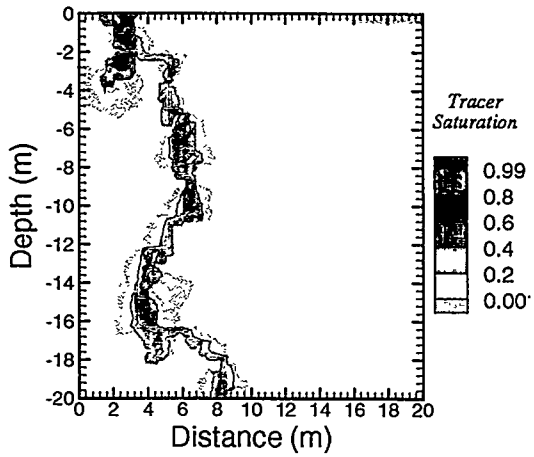
Figure VI.7 shows four snapshots of tracer transport, using the transient water flow field shown in Figure VI.6(a) as the initial condition. Again, tracer was injected uniformly over the entire top boundary at a constant rate of 10^{-3} kg/s, starting at the time when the water simulation is terminated. To save computation time, this tracer simulation was terminated when the flux ratio is approximately 0.95. Similar transport behavior as present in Figure VI.3 is also observed in Figure VI.7. Moreover, Figure VI.8(a) shows that the tracer BTC can be reasonably fitted by the log-normal model, though the tail of the analytical CDF deviates from the sample BTC. The deviation at the tail may be because the tracer transport has not reached the steady state yet. For comparison, Figure VI.8(b) shows the BTC obtained by injecting tracer to the steady state flow field of Figure VI.6(b). The resulting flow fields of tracer are not shown here because they are very close to those flow fields in Figure VI.7. Because the former tracer simulation was terminated at a higher flux ratio (at 0.97), the BTC in Figure VI.8(b) shows that it can be better fitted by the log-normal model. For the water BTC, however, Figure VI.9 shows that the log-normal distribution is not a good model for predicting water travel time.

VI. 2 Episodic infiltration

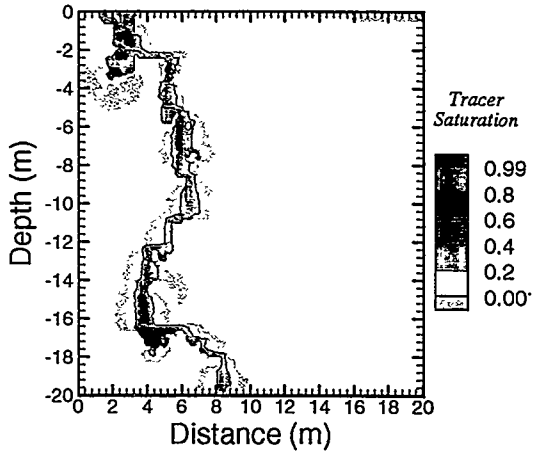
The study of episodic infiltration is motivated by field observations that in-situ surface infiltration into fractures may experience temporal variability to a large extent (Bodvarsson and Bandurraga, 1996). Under such conditions, fractures are undergoing repeated wetting and drying cycles. Therefore, hysteresis effects may become significant for episodic infiltration events. However, detailed analysis of hysteresis effects is beyond the scope of the present study. Instead, this section focuses on episodic infiltration events,



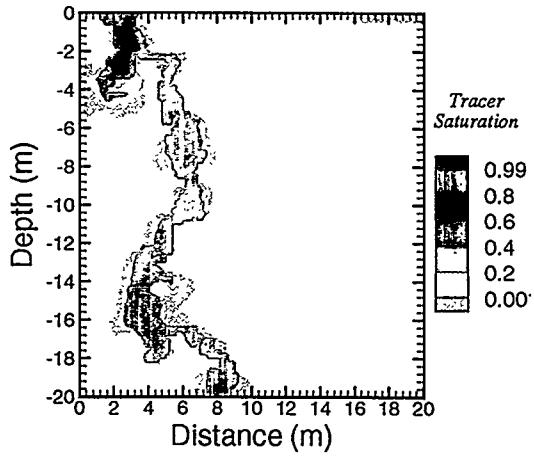
(a) $Q_{2,bot}/Q_{top} = 0.1$, Time = 0.39 days



(c) $Q_{2,bot}/Q_{top} = 0.9$, Time = 1.68 days

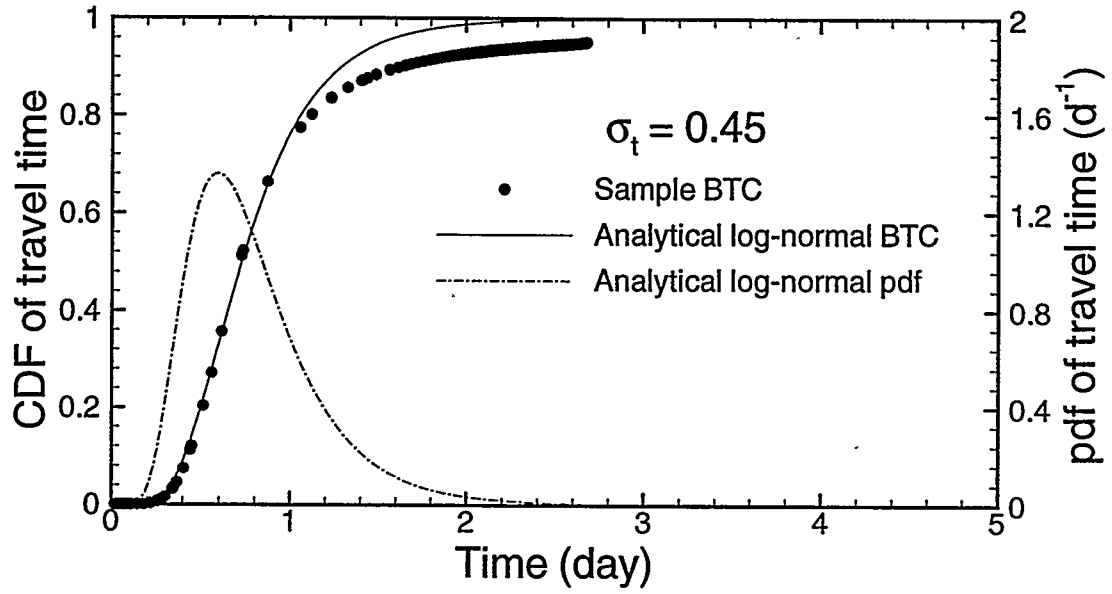


(b) $Q_{2,bot}/Q_{top} = 0.5$, Time = 0.74 days

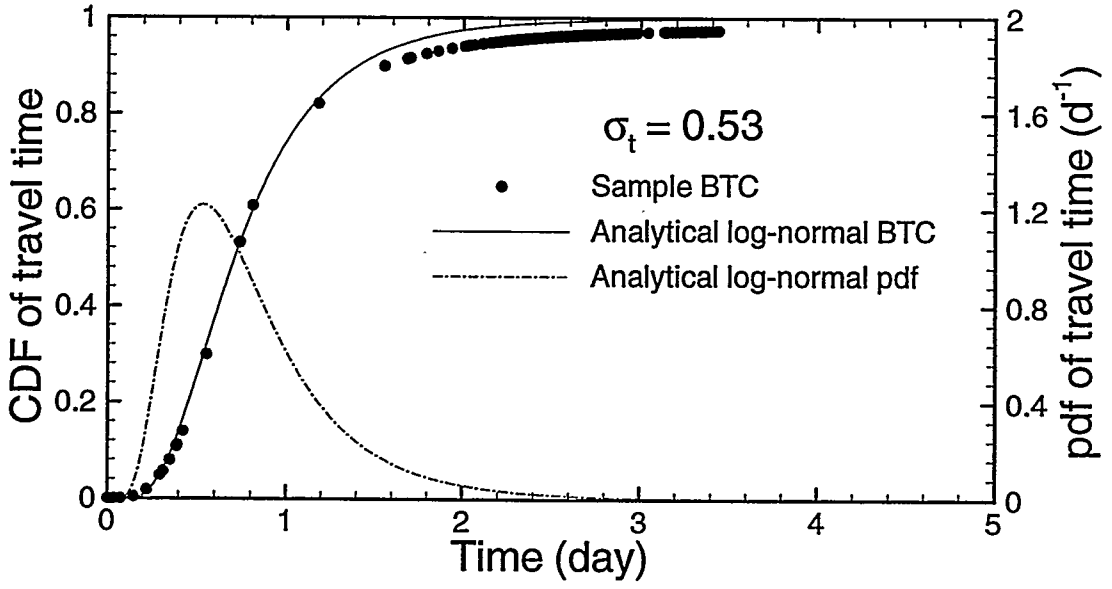


(d) $Q_{2,bot}/Q_{top} = 0.95$, Time = 3.44 days

Figure VI.7 Snapshots of tracer transport continued from the transient flow field in Figure VI.6(a).



(a) Tracer BTC from transient flow field of water



(b) Tracer BTC from steady state flow field of water

Figure VI.8 Two tracer BTCs corresponding to simulations with different initial conditions. (a) The initial condition is the transient flow field in Figure VI.6(a); and (b) The initial condition is the steady state flow field in Figure VI.6(b).

which are either events with repeated wetting and drying cycles or events with temporally intensified infiltration. The purpose of these analyses is to study the change of seepage patterns with respect to the change of infiltration rate applied at the boundary.

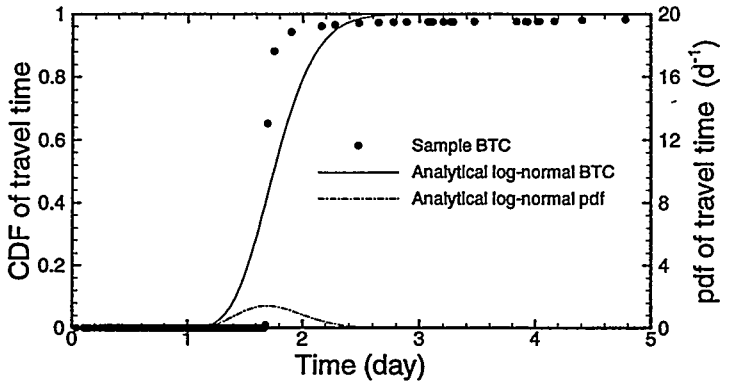
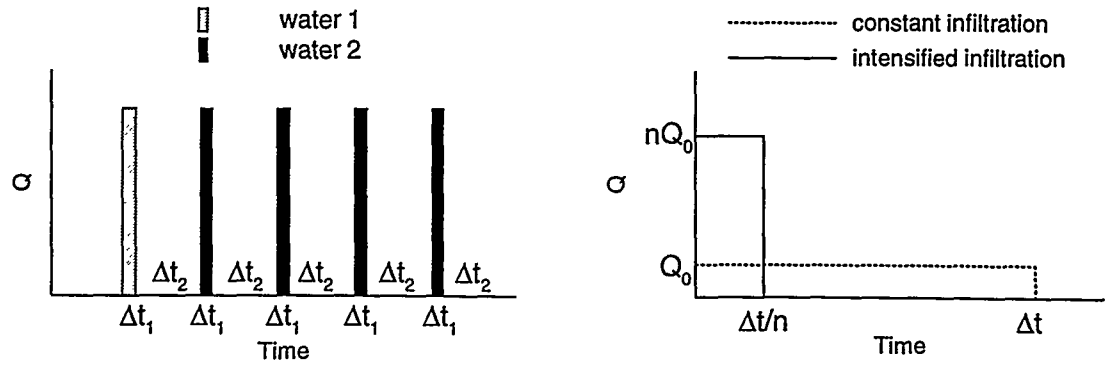


Figure VI.9 Water BTC corresponding to flow field in Figure VI.6(b).

Two types of episodic infiltration events are illustrated in Figure VI.10. The first type, Figure VI.10(a), considers a periodic infiltration event. The intervals of wetting and drying periods are Δt_1 and Δt_2, respectively. For the following example, the values of Δt_1 and Δt_2 are assumed to be 1 and 10 days, respectively. The infiltration rate for this example is still assumed to be 10^{-3} kg/s. Note that water 1 is the supplying fluid during the first wetting cycle, but it is changed to water 2 in subsequent wetting cycles. The purpose of switching the supplying fluid is to examine the effect of antecedent saturation history on seepage. The second type considers a transient infiltration event with a temporary intensified supply rate. This is illustrated in Figure VI.10(b) as the solid line. The interval Δt is the breakthrough time for constant supply rate of Q_0. The arbitrary integer n is the ratio of the intensified rate to the constant rate. Three ratios are considered, i.e., 2, 5, and 10. Note that the total amount of liquid injected into the fracture



(a) infiltration with repeated wetting and drying

(b) temporally intensified infiltration

Figure VI.10 Schematic of episodic infiltration events.

is the same, which is $Q_0 \times \Delta t$. In both scenarios, fluid is injected uniformly over the entire top boundary at a constant rate of 10^{-3} kg/s. The initial saturation at non-asperities is assumed at the value of residual water saturation.

Figure VI.11 shows the snapshots of the total water saturation at the end of each wetting and drying cycles for the first episodic infiltration event. For comparison, the realizations of water seepage at breakthrough and steady state for the corresponding single wetting event are shown at the top of the figure. For both wetting and drying cycles, water travels along the flow paths that are depicted in the steady state flow field of water. However, the speed of downward migration during a drying cycle is slower than the speed during a wetting cycle. This change of speed is shown in Figure VI.12 for the first wetting and drying cycles. The average speed of downward migration is reduced from 0.47 m/hr for the first wetting cycle to 0.31 m/hr for the first drying cycle. The reduction of speed is because of the lower relative permeability at the smaller liquid saturation during a drying cycle. The change of capillary pressure is shown in Figure VI.13(a) to VI.13(c).

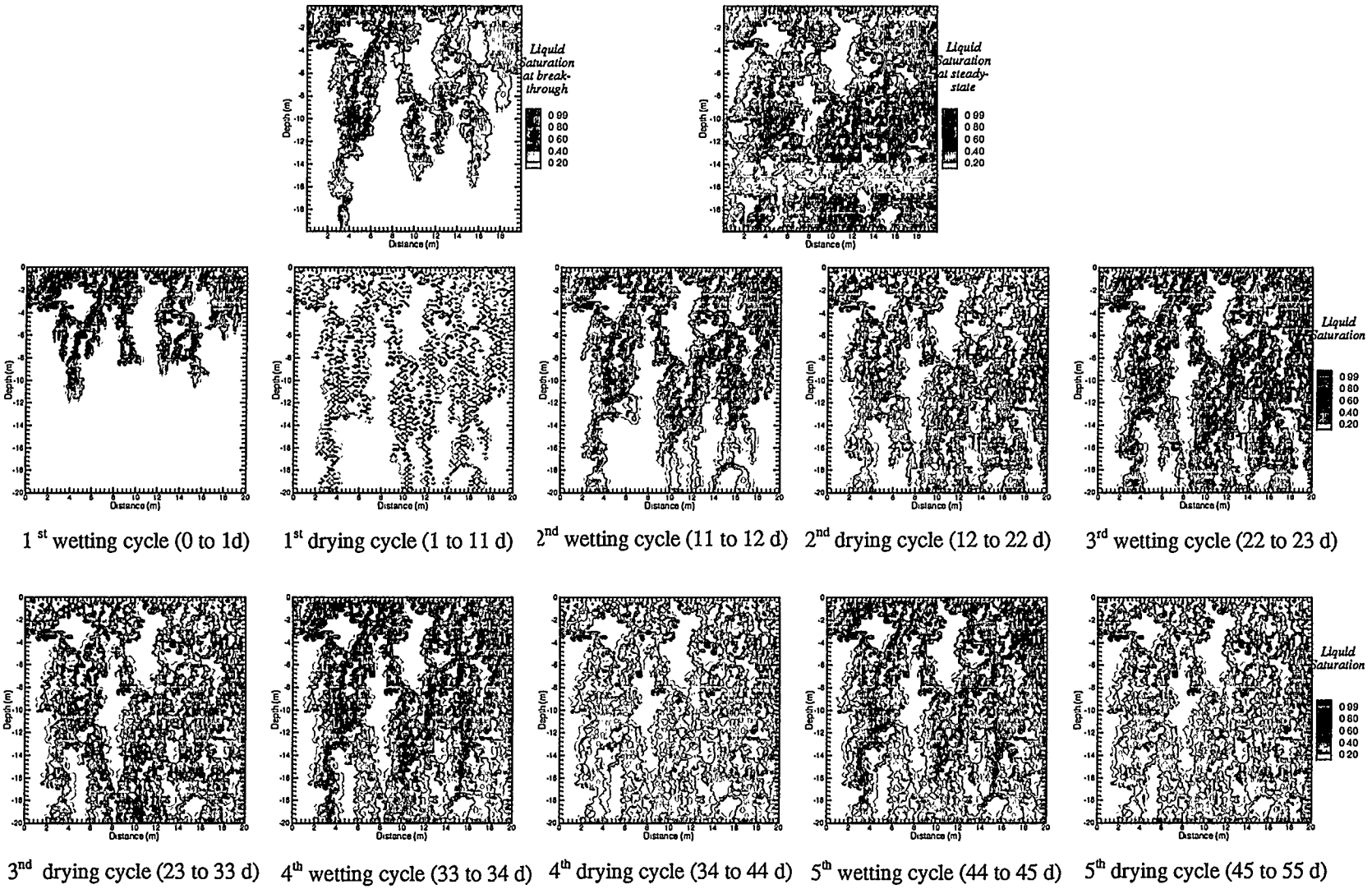


Figure VI.11 History of water saturation of five repeated wetting and drying cycles. Each wetting cycle lasts for one day with a constant injection rate of 10^{-3} kg/s, and drying cycle for 10 days. The top two plots show the liquid saturation of a single wetting cycle at breakthrough and steady state, respectively, with a continuous, constant injection of 10^{-3} kg/s.

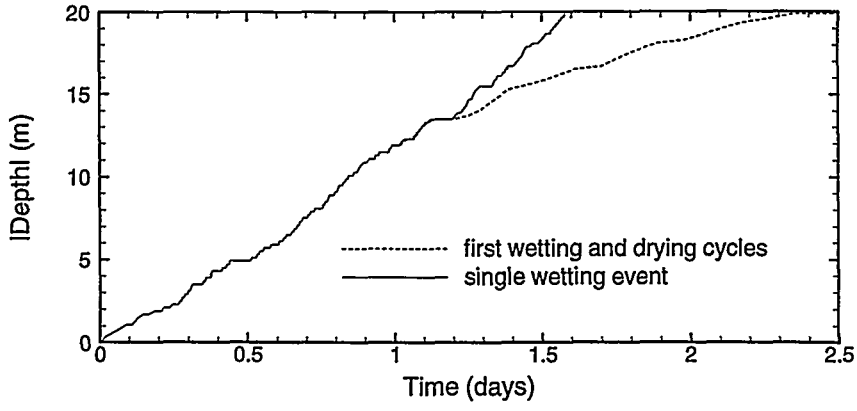


Figure VI.12 Vertical advance curves of the first wetting and drying cycles, and the single wetting event.

For subsequent wetting cycles, water only travels within flow paths that have been developed by previous drying cycle. This is because those flow paths have higher relative permeability than dry flow paths. New flow paths can only be developed during drying cycles. The reason is that, from capillary theory, water (the wetting phase) tends to remain in smaller pores during drying cycles. Since dry flow paths have smaller relative permeability to water than wet flow paths, their effective pore space is smaller than that for wet flow paths. Change of relative permeability in consecutive wetting and drying cycles can be seen in Figure VI.13(d) to VI.13(f). It is obvious that the relative permeability in a drying cycle is smaller that that in wetting cycles.

Because the governing Richards' equation does not consider hysteresis effects, liquid seepage in Figure VI.11 starts to repeat itself approximately after the third wetting-drying cycle. However, the flow fields near the lower boundary after the second drying cycle are somewhat different from each other because the wetting front is still sensitive to the surface infiltration. Simulation continues after the fifth drying cycle by constantly

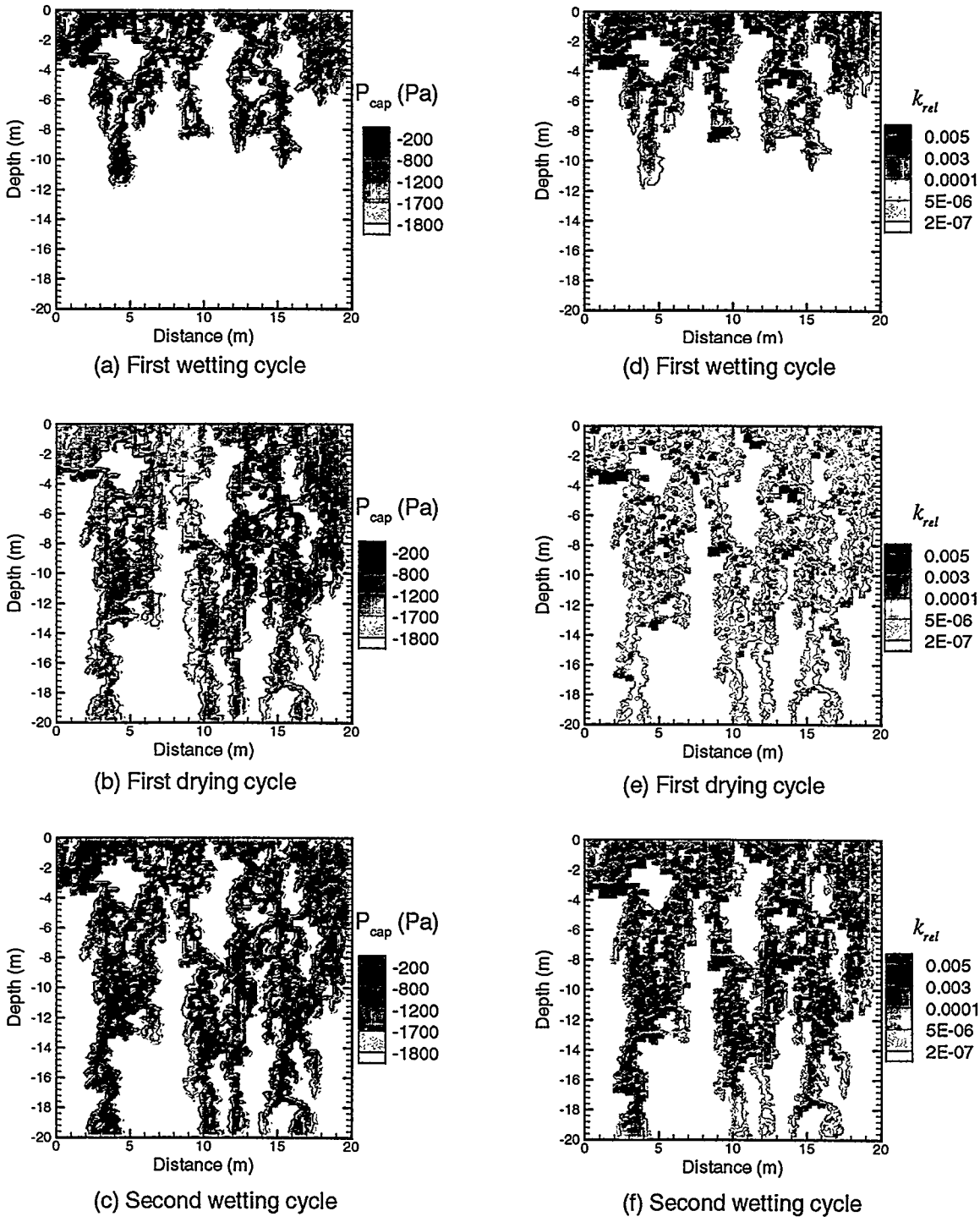


Figure VI.13. Capillary pressure and relative permeability for the first and second wetting cycles and the first drying cycle.

injecting water 2 into the fracture, and then stops when the flux ratio of water 2 is 0.933. The resulting flow field of total water is shown at the top left in Figure VI.14. This flow field is, as it should be, very close to the steady state flow field of water of the single wetting event because hysteresis effects are neglected. The top right plot in Figure VI.14 shows the water 2 saturation after the fifth drying cycle. For completeness, flow fields of water 2 after the first drying cycle are shown in the rest of Figure VI.14.

Figure VI.15 shows the periodic change of saturation at the top surface for the five wetting and drying cycles. For the first wetting-drying cycle, Figures VI.15(b) show that saturation at the top surface increases rapidly to a constant value during a wetting cycle and then decreases gradually to a lower value during a drying cycle. Because the capillary hysteresis is neglected, it is expected that the absolute capillary pressure at the top surface also experiences similar variations. That is, the flow simulation switches to a constant head boundary condition during a short interval, even though the simulation starts from a constant injection boundary condition. It is expected, therefore, that flow simulations using a constant head boundary would be close to the present simulations.

Because drying duration was only 10 days in previous simulations, significant changes of capillary pressure or other dynamic processes may not be evident. For example, Figure VI.16 shows that the transient change of accumulated mass at the bottom boundary behaves as an increasing function with time. If the drying duration was long enough, it would be expected to see this curve to be stabilized before the end of each drying cycle. On the other hand, the exiting flux of water 2 at the bottom boundary tends to stabilize at the end of each drying cycle, see Figure VI.17.

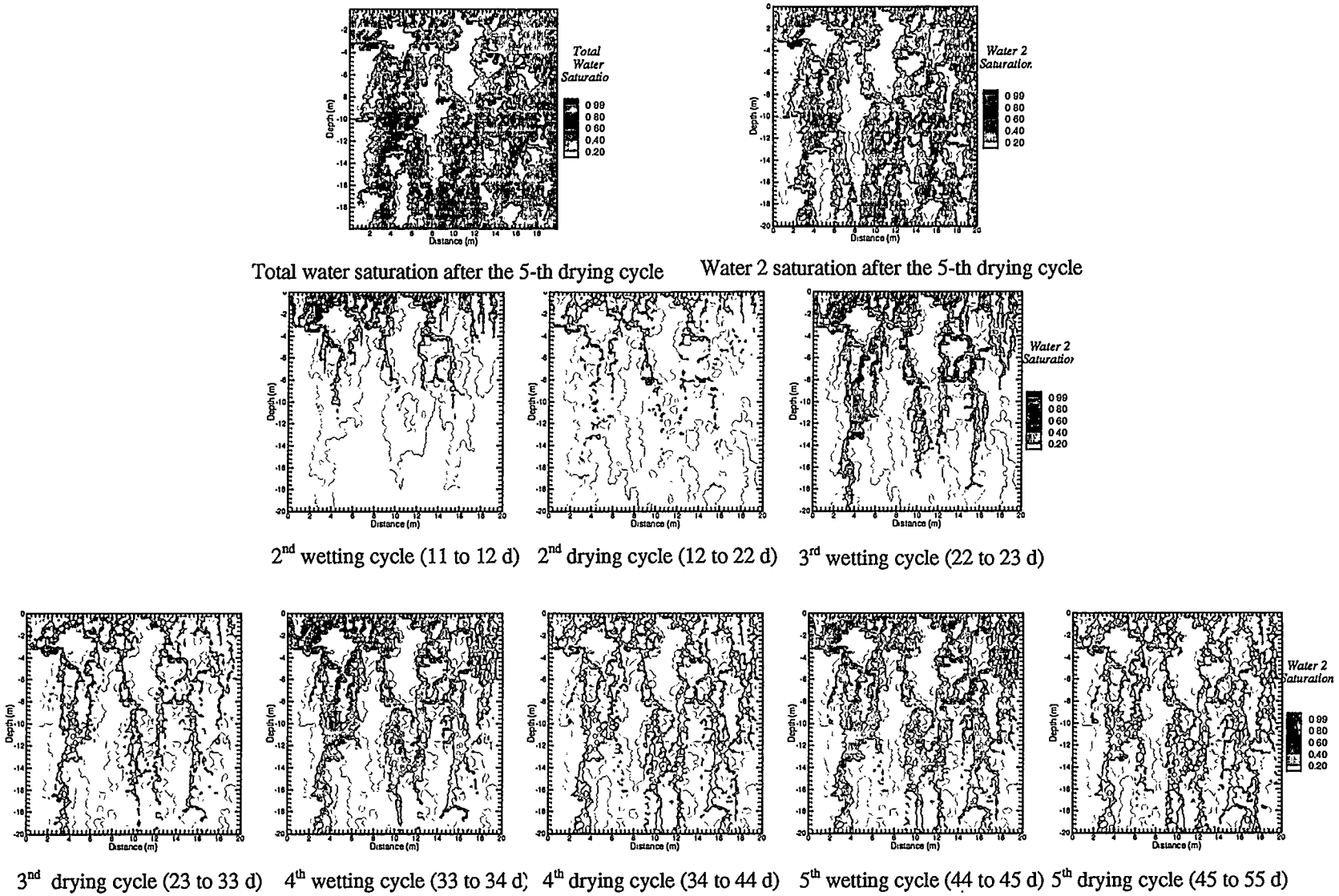


Figure VI.14 History of water 2 saturation of after the first drying cycle. The top two plots show the total water and water 2 saturation, respectively, for which water 2 is continuously injected into the fracture after the 5-th drying cycle. The flow fields for the top two plots are terminated at the time at when $Q_{2,bot}/ Q_{top} = 0.933$.

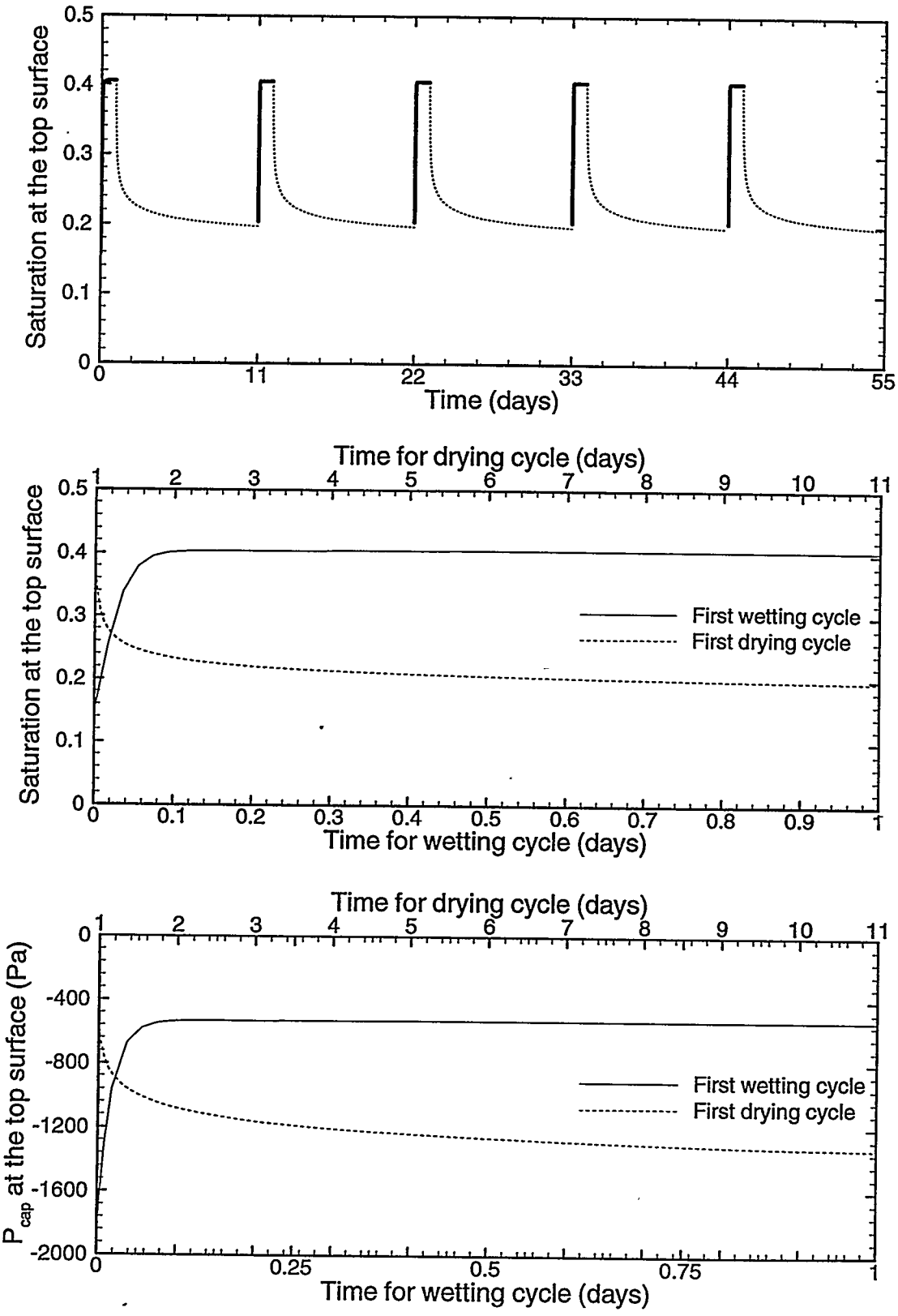


Figure VI.15 Change of saturation and capillary pressure at the top surface.

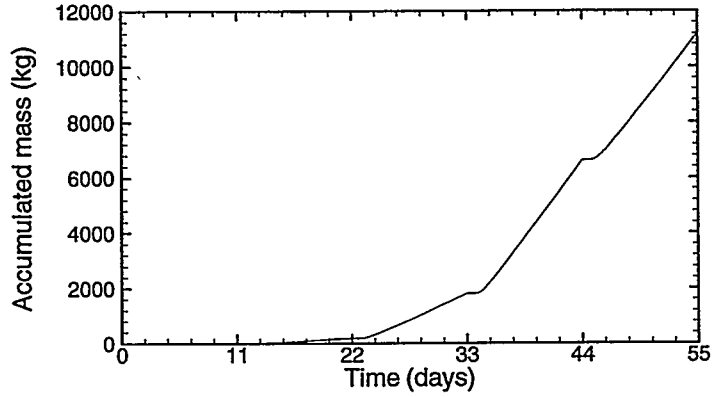


Figure VI.16 Change of accumulated mass of water 2 after the first drying cycle.

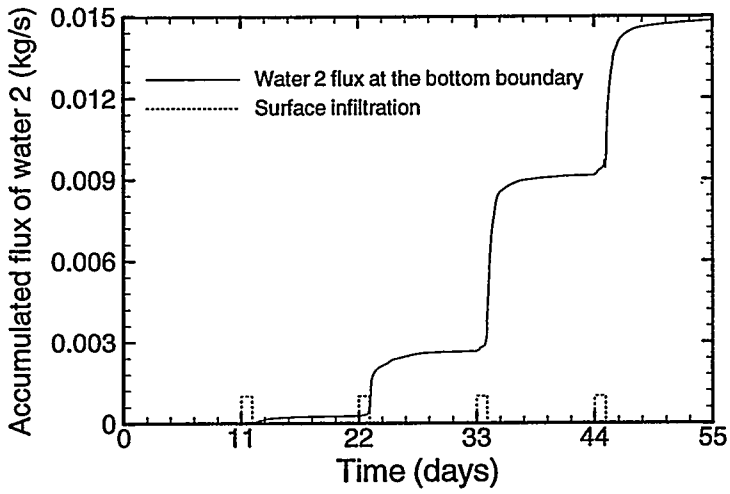


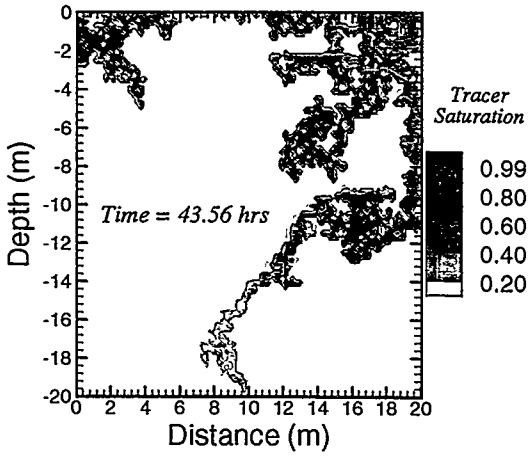
Figure VI.17. Change of accumulated water 2 flux at the bottom boundary.

Now, let us consider the second type of episodic infiltration. The constant supply rate (Q_0) and the time to breakthrough (Δt) are 10^{-3} kg/s and 43.56 hrs, respectively, for this numerical experiment. Three ratios of the flow rate (2, 5, and 10) are considered. Therefore, the new injection rates are 2×10^{-3} kg/s, 5×10^{-3} kg/s, and 1×10^{-2} kg/s, and the new intervals of injections are 21.78 hrs, 8.712 hrs, and 4.356 hrs.

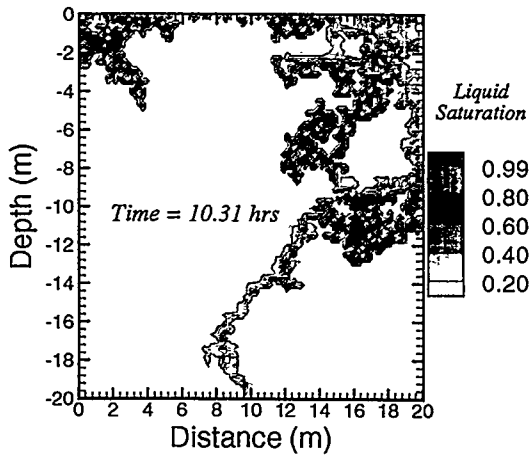
Figure VI.18 shows realizations of saturation at breakthrough for events with a uniformly distributed supply rate and intensified rates. All realizations are practically the same, only with a minor difference due to free drainage after surface infiltration is turned off. However, Figure VI.19 shows that the vertical seepage for all episodic infiltration events evolve at a faster speed than the event with a uniformly distributed supply rate. If the average seepage velocity is defined as the ratio of the depth of the plume tip to travel time, Table VI.1 (see column 4) shows that the initial average velocity increases with increasing surface infiltration. As surface infiltration increases, however, the ratio of average velocity for episodic infiltration to the average velocity for constant infiltration becomes smaller than the ratio of supply rate, see column 6 of Table VI.1. On the other hand, the ratio of ponding duration is approximately the inverse of the ratio of supply rate. Furthermore, liquid seepage after ponding proceeds at a faster speed than seepage before ponding, see the last column of Table VI.1 and Figure VI.18.

These simulation results suggest that liquid seepage for episodic infiltration follows some patterns if the total mass injected into fracture is conserved. These patterns can be roughly divided into three stages: before ponding, during ponding and after ponding. The ponding refers to the regions which significantly delay the seepage. For example, the ponding refers to the regions above the depth at – 13 m in Figure VI.18. It is obvious that these patterns are different from each other.

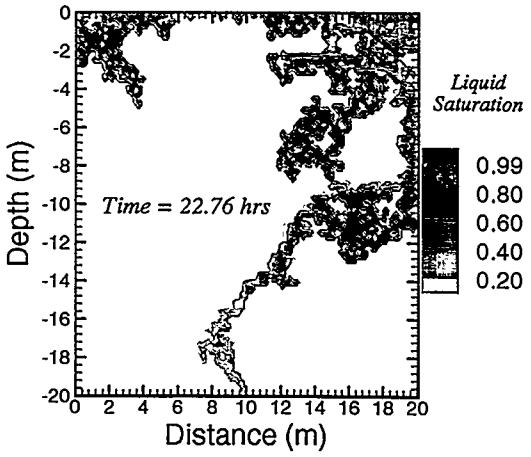
The first pattern applies to the time interval before the seepage develops significant ponding. It says that the initial average seepage velocity increases with increasing surface infiltration. This increasing trend is because the flow resistance



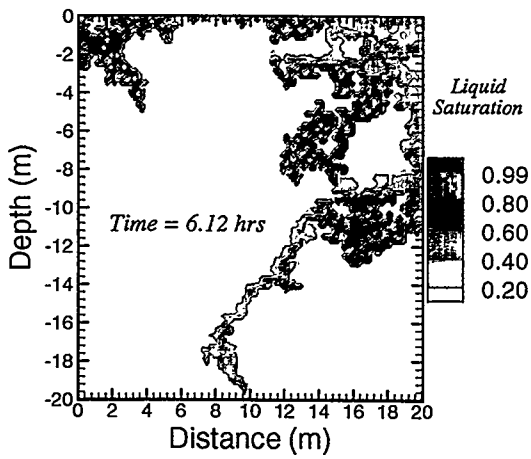
(1) Water saturation with $Q = 10^{-3}$ kg/s. Water is injected within the interval from 0 hr to 43.56 hrs.



(3) Water saturation with $Q = 5 \times 10^{-3}$ kg/s. Water is injected within the interval from 0 hr to 8.71 hrs.



(2) Water saturation with $Q = 2 \times 10^{-3}$ kg/s. Water is injected within the interval from 0 hr to 21.78 hrs.



(4) Water saturation with $Q = 1 \times 10^{-2}$ kg/s. Water is injected within the interval from 0 hr to 4.36 hrs.

Figure VI.18 Saturation for episodic infiltration events with intensified rates.

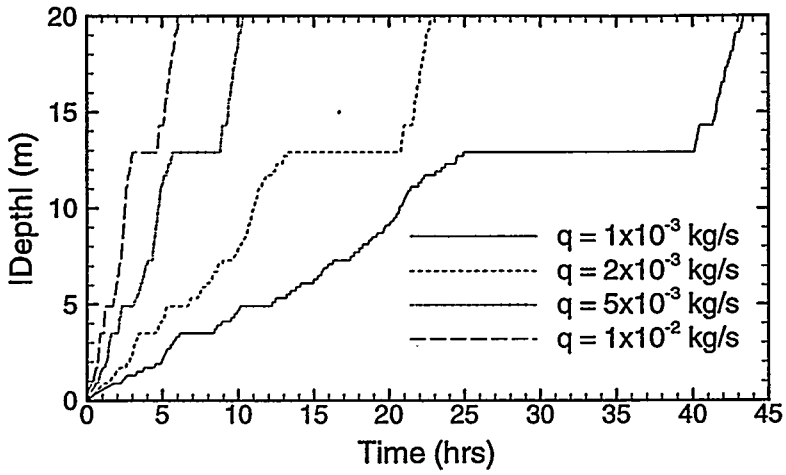


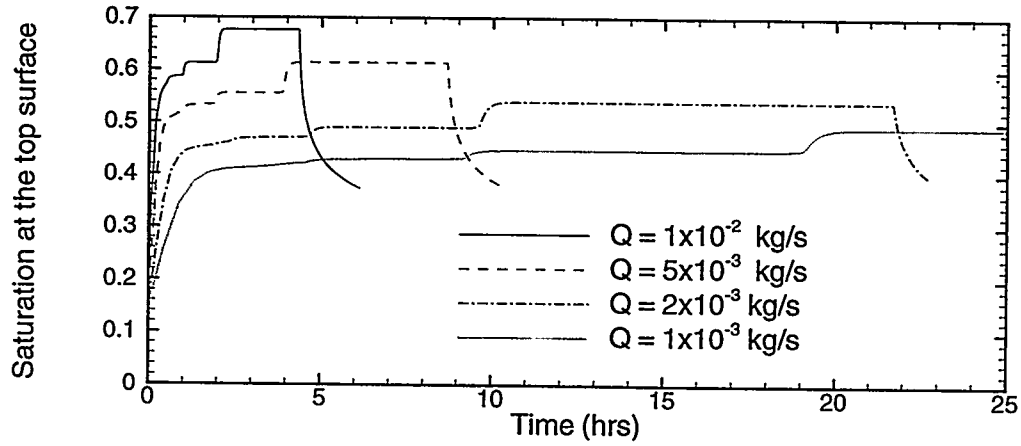
Figure VI.19 Vertical advance curves for episodic infiltration events.

Table VI.1 Results of episodic injection

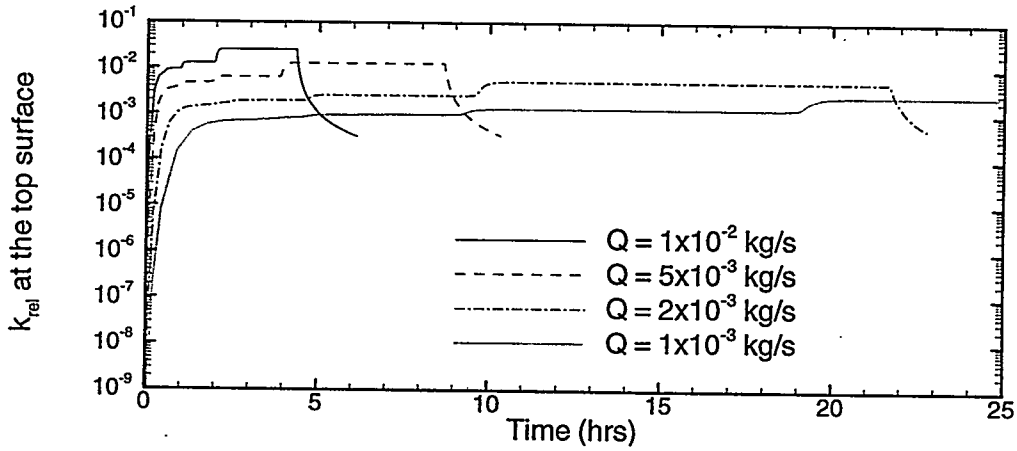
Q (kg/s)	T (hr)	T_f (hr)	V_f (m/hr)	R_Q	R_V	T_p(hr)	V_p(m/hr)
1×10^{-3}	43.56	24.93	0.52	1	1.00	15.17	3.17
2×10^{-3}	22.76	13.23	0.98	2	1.89	7.51	5.19
5×10^{-3}	10.31	5.69	2.27	5	4.37	3.11	8.81
1×10^{-2}	6.12	3.00	4.30	10	8.27	1.68	8.39

Q = surface infiltration rate
T = breakthrough time
T_f = first arrival time to the depth of – 13.0 m, the depth where ponding occurs
V_f = initial average seepage velocity, i.e., $13.0/T_f$
R_Q= ratio of surface infiltration, i.e., $Q/1 \times 10^{-3}$
R_V = ratio of the initial average seepage velocity for the episodic infiltration to the average velocity for the event with a uniformly distributed rate
T_p = time needed to completely saturate the ponded regions at –13.0 m
V_p = average seepage velocity after ponding

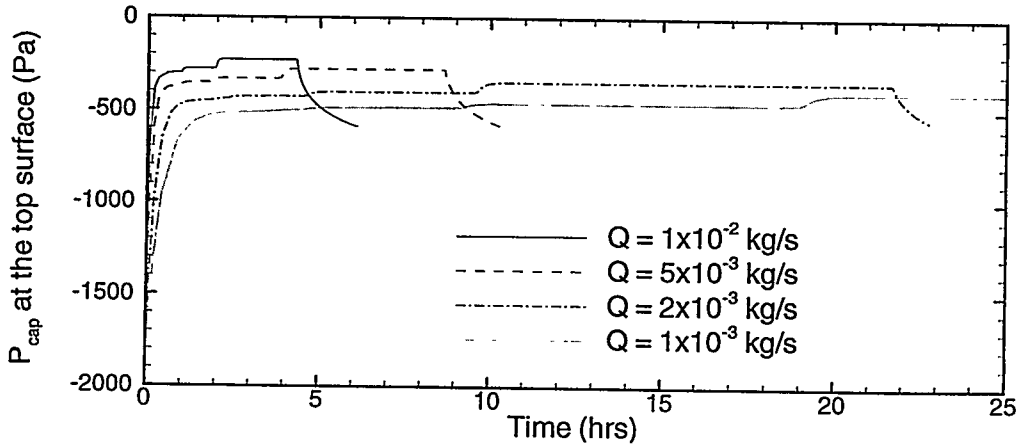
decreases with increasing supply rate. The decrease of flow resistance with respect to injection rate is interpreted by the increasing relative permeability and decreasing absolute capillary pressure at the top surface, see Figure VI.20 (b) and VI.20(c). This initial seepage velocity remains approximately constant until water descends to a depth at about –13 m at which significant ponding is occuring. However, the increase of the initial seepage velocity is not at the same pace as the increase of supply rate. This may be



(a) surface saturation



(b) relative permeability at the surface



(c) capillary pressure at the surface

Figure VI.20 Saturation, relative permeability and capillary pressure at the top surface for constant and sporadic infiltrations.

because seepage needs longer time to develop new flow paths as the supply rate increases. Thus, the ratio of seepage velocity for episodic infiltration to the velocity for the event with uniformly distributed supply rate is smaller than the ratio of supply rate.

Note that each of the curves in Figure VI.20 for episodic infiltration drops off because of the termination of surface infiltration. In contrast, the tails of curves for constant infiltration remain horizontal. In addition, all the curves in Figure VI.20 exhibit a stepwise increasing behavior. This is because liquid has to fill in dead-end pores before developing new flow paths. Thus, there are two obvious jumps in each of these curves. Each jump corresponds to each dead-end pore shown in Figure VI.18.

The second-stage pattern suggests that the time needed to completely saturate the ponded regions at – 13 m inversely follows the pattern of supply rate. This is simply because of mass balance. That is, the larger the supply rate, the shorter the duration to fill the pore space.

The third-stage pattern suggests that seepage after ponding has an average velocity that is even faster than the initial seepage velocity. However, there seems to be no explicit correspondence to the pattern of supply rate. But, results in Table VI.3 suggest that average seepage velocity after ponding becomes closer to each other as the supply rate increases.

In summary, these patterns observed in Figure VI.19 suggest that ponding duration inversely correspond to the pattern of surface infiltration. However, average seepage velocity follows different patterns before and after ponding. Combining these

patterns implies that ponding may have less impact if the surface infiltration becomes stronger. For an infiltration event with a high supply rate, liquid seepage may follow a single average seepage velocity even if significant ponding appears. On the other hand, if the surface infiltration become weaker, time lag due to ponding may have significant impact on liquid seepage as well as BTC behavior.

VI.3 Limitations of space and time averaging

It is known that the macroscale approach, instead of representing the spatial heterogeneity of fractures with a detailed resolution, averages the heterogeneity of fractures as well as the inputs to the system boundary. In this section, a numerical experiment is conducted to point out the limitations of such an approach.

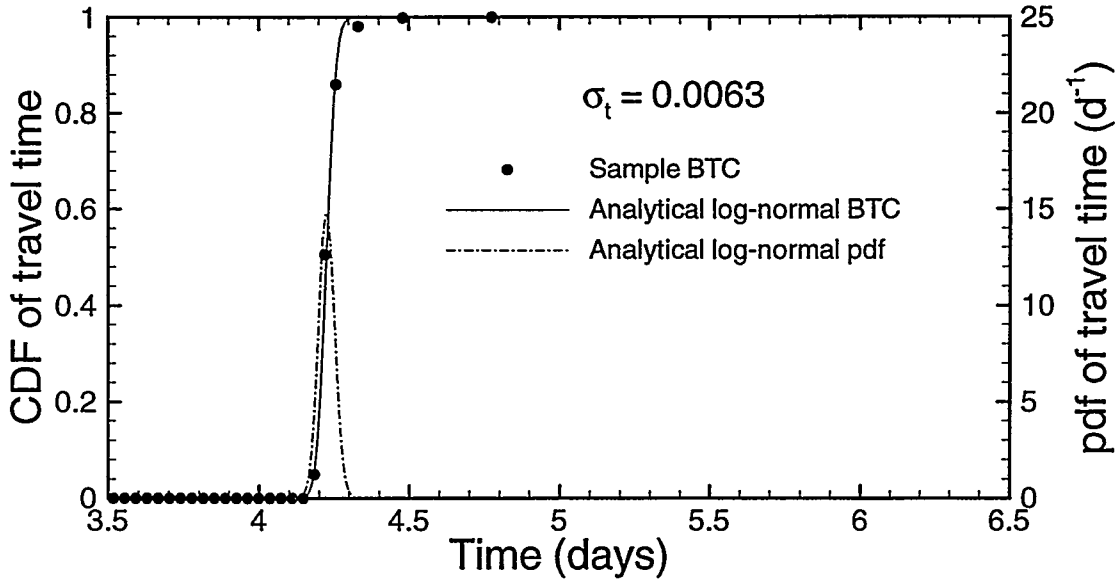
Consider, for example, the synthetic fracture in Figure V.6(d). This fracture has spatially random asperity contacts with a total volumetric fraction of 40%. To simulate the macroscale averaging approach, the heterogeneities are replaced with spatially averaged porosity and effective permeability. The same initial and boundary conditions as those specified in section V.1 are used for the simulation. The resulting values of parameters for this flow simulation are listed in Table VI.2. The computed saturation at breakthrough is spatially uniform with a value of 0.58. In addition, the vertical advance curve for transient flow field is perfectly linear. However, the time to breakthrough obtained by the macroscale approach is longer than that obtained using detailed resolution of fracture heterogeneity. After the first breakthrough, however, the liquid seep corresponding to the macroscale approach takes a shorter time to reach the steady state. Overall, the macroscale averaging approach is not able to simulate the occurrence of fast

preferential flow paths as observed in the fields. More importantly, the macroscale approach cannot predict the complicated seepage patterns, such as fingering, bypassing and ponding, that are expected to be seen in the fields.

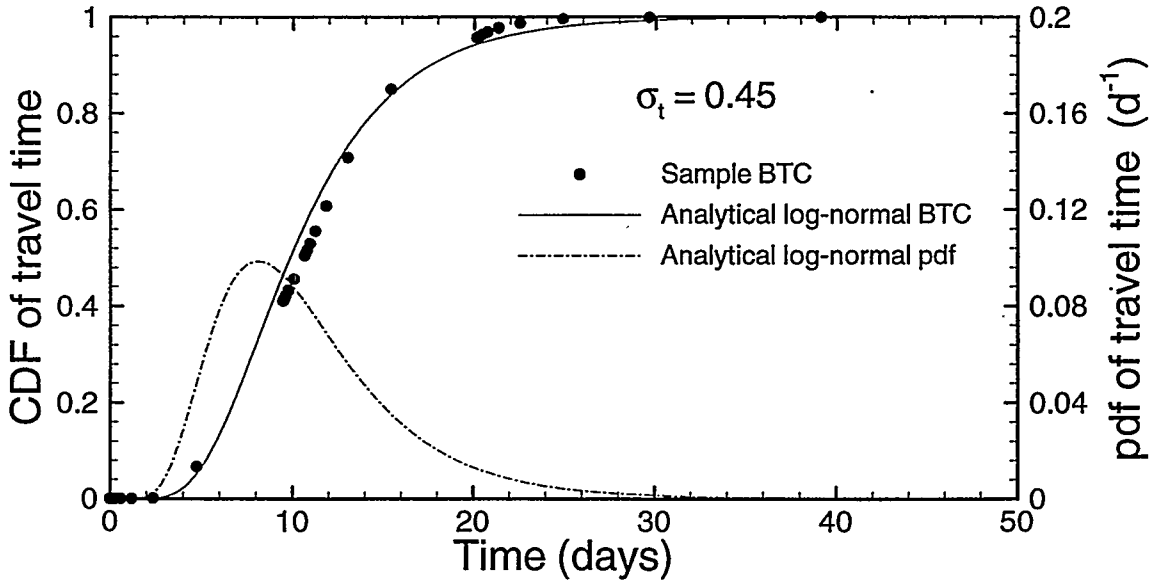
Table VI.2 Comparisons between the macroscale approach and the current approach

Simulation method	ϕ	$k_{eff,h}$ or k (m^2)	$k_{eff,v}$ or k (m^2)	Breakthrough time (hrs)	Steady state time (days)
Macroscale approach	0.21	0.055×10^{-9}	0.060×10^{-9}	98.64	4.78
Current approach	0.35	$1.0 \times 10^{-9} \times \zeta$	$1.0 \times 10^{-9} \times \zeta$	34.85	18.80

The water and tracer BTCs for the simulation using the macroscale approach are shown in Figure VI.21. Tracer simulation is continued from the steady state flow field obtained by the macroscale approach. The same boundary conditions as those used for the water simulation are employed for the tracer simulation. Again, the resulting tracer flow field is trivial because it is also a spatially uniform saturation field. Figure VI.21 shows that the log-normal model can be fitted to both water and tracer BTCs. Note that the σ_t for water BTC is very small ($\sigma_t = 0.0063$). Thus, the water BTC can be practically approximated by a step function, and its pdf can be represented by a spike at the mean travel time ($<t> = 4.22$ days). However, tracer BTC is more dispersed than water BTC ($\sigma_t = 0.45$).



(a) Water BTC



(b) Tracer BTC

Figure VI.21 Water and tracer BTCs for simulations using the macroscale averaging approach.

Chapter VII. Discussion and conclusions

Field evidence suggests that in semi-arid environments water is able to migrate downward rather rapidly along preferential paths. For example, at Yucca Mountain, Nevada, environmental tracers have been shown to migrate several hundred meters deep within decades. However, the time scale of tracer transport predicted by numerical models using conventional volume-averaging approaches is on the order of thousands of years. In order to address this discrepancy, a numerical model based on fundamental processes and mechanisms has been proposed in this study.

Attempts at modeling flow and transport in unsaturated fractured rock based on a mechanistic process model must start from a specification of void space geometry in fractures. Unfortunately, the multiple length scales of fracture surfaces generally complicates the specification of void space geometry. On the other hand, our ability to directly obtain geometric characteristics of fracture void spaces from field observations is limited. Only inputs into and outputs from the flow system boundaries can be observed in the field, which are all subject to significant temporal as well as spatial variability. In this study, void spaces in fractures are characterized based on hypothesized geometric features, such as spatially correlated asperity contacts. These spatial characteristics are expected to be most relevant for seepage behavior. The appropriateness of these geometric features is then judged by whether they are able to reproduce flow and transport behavior that would be observed in the field.

Fluid flow and solute transport in natural fractures generally occur in 3-D fracture networks. In this study, however, fracture networks were approximated as 2-D

heterogeneous porous media in a vertical fracture plane. In addition, the effect of matrix permeability was neglected. This assumption is reasonable for short-term flow and transport behavior, but may not be viable when considering long-term flow patterns. Thus, the present study is applicable to "small" fractures in hard rocks of low permeability, such as welded tuffs, graywacke, mudstones, granite, and some fractured basalts. It would not be applicable to larger fractures with 3-D void space, or to small fractures in rocks with significant matrix permeability (e.g., non-welded tuffs or sandstones).

Several approximations and assumptions were made in this study. Effects of entrapped air were neglected. Hysteresis effects in capillary pressure and relative permeability were also neglected. Furthermore, permeability heterogeneity was assumed to be the dominating influence on seepage. Porosity heterogeneity was not considered. For solute transport, molecular diffusion was neglected. The last assumption suggests that the only mixing mechanism is due to the finite spatial discretization. In addition, neglecting molecular diffusion limits the time scale for which the realistic solute transport can occur.

Among the various spatial features of fracture void spaces, the spatial correlation around asperity contacts is focused in this study. This is motivated by preliminary analyses that conventional semi-variograms are not very sensitive to the topology of asperity contacts in fractures. The reason for this insensitivity may be because the detailed heterogeneity of a random field is averaged out by the semi-variogram. Thus, a modified Metropolis algorithm is proposed as a new perturbation mechanism for

simulated annealing (SA). This modified algorithm is able to emphasize the neighborhoods of conditioning asperity contacts, while formulating the objective function by employing the conventional semi-variogram. Simulated permeability fields obtained by the modified Metropolis algorithm thus show much stronger clustering of asperity contacts.

It was found from sensitivity analyses that the ratio of $\Delta/\lambda_k < 0.5$ is preferred for better representation of spatially correlated permeability fields. However, seepage patterns were not as sensitive as characterization accuracy to this ratio. Indeed, seepage patterns are virtually insensitive to the ratio Δ/λ_ζ as long as its is smaller than 0.5. This is especially true for flow simulations in permeability fields with spatially correlated asperity contacts. This insensitivity is explained by the significant bypassing effect of flow due to asperity contacts that are laterally correlated to a large extent.

Seepage in unsaturated fractures with either localized or distributed injection is characterized by localized preferential flow, along with bypassing, funneling, and localized ponding. Generally, flow and transport behavior is dominated by the fraction of asperity contacts, and their shape, size, distribution and spatial correlation. However, the detailed distribution of permeability in the open space of fracture is less important than the spatial correlation of asperity contacts. For increasing fraction of asperity contacts, there is more flow bypassing and ponding, but fewer fingers. For a fixed fraction of asperity contacts, however, flow bypassing, fingering and average vertical seepage velocity depend on the correlation length and the principal correlation direction of asperity contacts. If asperity contacts are horizontally correlated, flow bypassing,

fingering, and average vertical seepage velocity generally increase with increasing horizontal correlation length of asperity contacts. For non-horizontally or non-vertically correlated asperity contacts, flow bypassing and average vertical seepage velocity increase as the anisotropy ratio of asperity contacts increases; but flow fingering decreases with increasing anisotropy ratio.

Seepage dispersion is generally higher for fractures with intermediate fraction of asperity contacts; but it is lower for small or large fractions of asperity contacts. The reason for this behavior is that many fingers can be formed in fractures with small fraction of asperity contacts. These fingers are not significantly affected by asperity contacts, and they all have similar breakthrough behavior. Thus, seepage dispersion is weak. With a large fraction of asperity contacts, only a limited number of fingers (sometimes only one) can break through; thus, seepage dispersion is generally weak. However, a few fingers (both fast and slow) are formed in fractures with intermediate fraction of asperity contacts. Thus, seepage dispersion is generally stronger.

Ponding occurs in regions that lack permeability in the vertical direction. It is then expected that ponding would slow down the downward advancement of seepage. As a result, seepage dispersion may become larger because of ponding, even for fractures with large fraction of asperity contacts. However, if ponding is significant, it may gather distributed seepage and focus flow into more localized pathways. Accordingly, seepage may be accelerated because the funneled flow has a higher seepage velocity. Under such circumstances, seepage dispersion may be greatly reduced, and the resulting water BTC behaves like a step function.

The strength of ponding depends on the fraction of asperity contacts and their correlation structure, i.e., their correlation lengths, anisotropy ratio, and the principal direction. In synthetic fractures with isotropic permeability and asperity contacts, the strength of ponding increases with increasing fraction of asperity contacts. Yet, the relationship between the strength of ponding and the anisotropic structure of permeability and asperity contacts is still not clear.

Simulation results found that BTCs for solute transport tends to be more dispersive than water BTCs. This is because water has to fill dead-end pores along its flow paths while migrating downward. However, tracers bypass these dead-end pores and travel along flow paths that have less flow resistance. Therefore, tracer transport is more uniform than water transport. More importantly, it was found that the log-normal travel time model does not fit water BTCs very well. In contrast, BTCs of solute transport either under transient or steady state flow field of water can be fitted very well by the log-normal model. The positively skewed log-normal distribution implies that tracer transport may evolve over a longer time scale than water transport.

The general features of flow patterns, as well as the different scaling laws with respect to infiltration events with different rates of surface injection have several implications for field experiments. First, the in-situ sampling techniques need to consider the effect of preferential flow and flow bypassing. Installing sampling devices in a spatially uniform manner may not be cost-effective because flow may only break through certain locations at the exiting boundary. Second, accelerated or delayed seepage due to ponding needs to be considered when designing the sampling intervals, especially for

automatic sampling equipment. Third, field experiments should be carefully designed to consider the temporal and spatial variability of the input to fractures. Results of our simulation suggest that initial seepage velocity increases with the infiltration rate. In addition, the effect of ponding also changes with the supply rate. Therefore, seepage patterns are also subject to change with the surface infiltration. For waste isolation problems, it may be necessary to locate the fast preferential flow paths to prevent the infiltrating water from contacting the storage canister. From the aspect of waste management, it may be needed to consider the transport pathways of solute once a storage canister is exposed to water.

In the future, studies of flow and solute transport in natural fractures should focus on employing more realistic assumptions. The numerical model should be expanded to consider 3-D effects. In addition, flow and solute transport in unsaturated fractures is at least a two-phase process, effects of the gas phase should not be neglected. For example, Richards' equation implicitly assumes that the non-wetting phase fluid (air/gas) does not interfere with the movement of the wetting phase. In reality, however, air may be trapped within dead-end pores or be accumulated ahead of a critical pore neck. Therefore, it may block the movement of the wetting phase. Its pressure may be increased to a critical point, e.g., the bubbling pressure, such that it may be released by bubbling or be pushed through the pore neck. Thus, phase interference and phase change are essential mechanisms for two-phase problems. The appearance of pore necks then raises the concern of the spatial variability of porosity. Porosity heterogeneity may have long-term effects on seepage as well as solute transport. Furthermore, the effect of matrix permeability should be included when considering long-term seepage effects. As a result,

molecular diffusion should also be considered because it is the dominating mechanism of mass transfer between the rock matrix and fractures. Furthermore, field observations indicate that surface infiltration and percolation are both subject to temporal variability. Seepage is therefore expected to experience hysteresis effects. Therefore, hysteresis effects of capillary and relative permeability should also be included.

References

Abdel-Salam, A., and C.V. Chrysikopoulos, Unsaturated Flow in a Quasi-Three-Dimensional Fractured Medium with Spatially Variable Aperture, *Water Resource Research*, Vol. 32, No. 6, pp. 1531-1540, 1996.

Abelin, H., L. Birgersson, J. Gidlund, L. Moreno, I. Neretnieks, S. Tunbrant, Results from some tracer experiments in crystalline rocks in Sweden. In: Tsang, C. F. (Ed.), Coupled Processes Associated with Nuclear Waste Respositories. Academic Press, New York, 1987.

Ahler, C. F., C. Shan, C. Haukwa, A. J. B. Cohen, and G. S. Bodvarsson, Calibration and prediction of pneumatic response at Yucca Mountain, Nevada using the LBNL/USGS three-dimensional, site-scale model of the unsaturated zone, Yucca Mountain Project Milestone OB12M, Lawrence Berkeley National Laboratory, Berkeley, CA, 1996.

Bear, J., *Dynamics of Fluids in Porous Media*, Elservier, New York, 764pp, 1972.

Bodvarsson, G. S., and T. M. Bandurraga, (Eds.), Development and calibration of the three-dimensional site-scale unsaturated-zone model of Yucca Mountain, Nevada, Yucca Mountain Site Characterization Project Milestone OBO2, Lawrence Berkeley National Laboratory Report LBNL-39315, Lawrence Berkeley National Laboratory, 1996.

Brace, W. F., Permeability of crystalline and argillaceous rocks, *Int. J. Rock Mech. Min. Sci. & Geomech. Abstr.*, Vol. 17, pp. 241-251, 1980.

Brown, S. R., and C. H. Scholz, Broad bandwidth study of the topography of natural rock surfaces, *J. Geophys. Res.*, Vol. 90, No. B14, pp. 12575-12582, 1985a.

Brown, S. R., and C. H. Scholz, Closure of random elastic surfaces in contacts, *J. Geophy. Res.*, Vol. 90, pp. 5531-5545, 1985b.

Brown, S. R., and C. H. Scholz, Closure of rock joints, *J. Geophy. Res.*, Vol. 91, pp. 4939-4948, 1986.

Brown, S. R., R. L. Kranz, and B. P. Bonner, Correlation between the surfaces of natural rock joints, *Geophysical Research Letters*, Vol. 13, No. 13, pp. 1430-1433, 1986.

Brown, S. R., Fluid flow through rock joints: the effect of surface roughness, *J. Geophys. Res.*, Vol. 92, pp. 1337-1347, 1987.

Brown, S. R., Simple mathematical model of a rough fracture, *J. Geophys. Res.*, Vol. 100, No. B4, pp. 5941-5952, 1995.

Chesnut, D. A., Characterizing the altered zone at Yucca Mountain: The beginning of a testing strategy, Proceedings, Third High Level Radioactive Waste Management International Conference, Las Vegas, NV, Vol. 1, pp. 1026-1039, American Nuclear Society, La Grange Park, IL, April 12-16, 1992.

Chesnut, D. A., Dispersivity in heterogeneous permeable media, Proceedings, Fifth Annual International High Level Radioactive Waste Management Conference, Las Vegas, NV, Vol. 4, pp. 1822-1841, American Nuclear Society, La Grange Park, IL, May, 1994.

Coakley K., K. Muralidhar, J. C. S. Long, and L. R. Myer, Equivalent permeability of statistically simulated single fractures, Proceedings of the conference on Geostatistical, sensitivity and uncertainty methods for ground-water flow and radionuclide transport modeling, edited by B. E. Buxton, pp. 441-469, 1987.

Copty, N., and Y. Rubin, A stochastic approach to the characterization of lithofacies from surface seismic and well data, *Water Resources Research*, Vol. 31, No. 7, pp. 1673-1686, 1995.

Dagan, G., P. Indelman, and G. Moltyaner, Stochastic analysis of concentration measurements in the transport experiment at Twin Lake site, *Water Resources Research*, Vol. 33, No. 4, pp. 559-567, 1997.

Datta-Gupta, A., L. W., Lake, and G. A. Pope, Characterizing Heterogeneous Permeable Media with Spatial Statistics and Tracer Data Using Sequential Simulated Annealing, *Mathematical Geology*, Vol. 27, No. 6, pp. 763-787, 1995.

de Marsily, G., *Quantitative Hydrogeology, Groundwater Hydrology for Engineers*, 440pp., Belmont. Academic Press, INC., San Diego, 1986.

Desbarats, A. J., Numerical estimation of effective permeability in sand-shale formations, *Water Resources Research*, Vol. 32, No. 2, pp. 273-286, 1987.

Desbarats, A. J., Macrodispersion in sand-shale sequences, *Water Resources Research*, Vol. 26, No. 1, pp. 153-163, 1990.

Deutsch, C. V., and A. G. Journel, GSLIB, Geostatistical Software Library and User Guide, Oxford University Press, New York, 340p., 1992.

Deutsch, C. V., and A. G., Journel, The Application of Simulated Annealing to Stochastic Reservoir Modeling, Advanced Technology Series, SPE, Richardson, Texas, Vol. 2, No. 2, pp. 222-227, 1994.

Deutsch, C. V., and P. W., Cockerham, Practical Considerations in the Application of Simulated Annealing to Stochastic Simulation, *Mathematical Geology*, Vol. 26, No. 1, pp. 67-82, 1994.

Doughty, C., Numerical modeling of field tests in unsaturated fractured basalt at the Box Canyon, Proceedings of the TOUGH Workshop 98', Ed., K. Pruess, Lawrence Berkeley National Laboratory Report, LBNL-41995, May, 1998.

Dougherty, C. R., and R. A., Marryott, Optimal Groundwater Management 1. Simulated Annealing, *Water Resources Research*, Vol. 27, No. 10, pp. 2493-2508, 1991.

Eaton, R. R., C. K. Ho, R. J. Glass, M. J. Nicholl, and B. W. Arnold, Three-dimensional modeling of flow through fractured tuff at Fran Ridge, Sandia National Laboratories, Albuequerque, NM, 1996.

Fabryka-Martin, J. T., A. V. Wolfsberg, P. R. Dixon, S. Levy, J. Musgrave, and H. J. Turin, Summary report of chlorine-35 studies: sampling, analysis and simulation of chlorine-36 in the Exploratory Studies Facility, Los Alamos National Laboratory Report LA-CST-TIP-96-002, Loa Alamos National Laboratory, Los Alamos, NM, August, 1996.

Faybishenko, B. A., Hydraulic behavior of quasi-saturated soils in the presence of entrapped air: Laboratory experiments, *Water Resources Research*, Vol. 31, No. 10, pp. 2421-2435, 1995.

Flint, L. E., Matrix properties of hydrogelogic units at Yucca Mountain, Nevada, US Geological Survey Open-File Report, US Geological Survey, Denver, CO, 1997.

Gauthier, J. H., M. L. Wilson, and F. C. Lauffer, Estimating the consequences of significant fracture flow at Yucca Mountain. Proceedings of the Third Annual International High-Level Radioactive Waste Management Conference, Las Vegas, NV, Vol. 1, pp. 891-898, American Nuclear Society, La Grange Park, IL, April, 1992.

Gauthier, J. H., An updated fracture flow model for total-system performance asessment of ucca Mountain, Proceedings of the Fifth Annual International High-Level Radioactive Waste Management Conference, Las Vegas, NV, Vol. 3, pp. 1663-1670, American Nuclear Society, La Grange Park, IL, April, 1994.

Glover, P. W. J., K. Matsuki, R. Hikima, and K. Hayashi, Synthetic rough fractures in rocks, *J. Geophys. Res.*, Vol. 103, No. B5, pp. 9609-9620, 1998a.

Glover, P. W. J., K. Matsuki, R. Hikima, and K. Hayashi, Fluid flow in synthetic rough fractures and application to the Hachimantai geothermal dry rock test site, *J. Geophys. Res.*, Vol. 103, No. B5, pp. 9621-9635, 1998b.

Isaaks, E. H., and R. M. Srivastava, Applied geostatistics, Oxford University Press, NY, 561pp., 1989.

Iwai, K., Fundamental studies of fluid flow through a single fracture, Ph. D. thesis, 208 pp., Univ. of Calif., Berkeley, 1976.

Johnson, D. S., C. R., Argon, L. A., McGeoch, and C., Schevon, Optimization by Simulated Annealing: An Experimental Evaluation, Part I, Graph Partitioning, *Operations Research.*, Vol. 37, pp. 865-892, 1989.

Johnson, N. M., and S. J. Dreiss, Hydrostratigraphic interpretation using indicator geostatistics, *Water Resources Research*, Vol. 25, No. 12, pp. 2501-2510, 1989.

Journel, A. G., Nonparametric estimation of spatial distributions, *Math. Geol.*, Vol. 15, No. 3, pp. 445-468, 1983.

Journel, A. G., and C. J. Huijbregts, Mining Geostatistics,600pp., Academic Press, London, 1978.

Kirkpatrick, S., C. D. Gelatt, and Jr. M. P. Vecchi, Optimization by simulated annealing, *Science*, Vol. 220, No. 4598, pp. 671-680, 1983.

Kitanidis, P. K., Introduction to geostatistics: Applications to Hydrogeology, 249pp., Cambridge University Press, NY, 1997.

Kranz, R. L., A. D. Frankel, T. Engelder, and C. H. Scholz, The permeability of whole and jointed Barre granite, *Int. J. Rock Mech. Min. Sci. Geomech. Abstr.*, Vol. 16, pp. 225-234, 1979.

Kung, K. J. S., Preferential flow in a sandy vadose zone: 1. Field observation, Geoderma, Vol. 46, pp. 51-58, 1990a.

Kung, K. J. S., Preferential flow in a sandy vadose zone: 2. Mechanism and implications, Geoderma, Vol. 46, pp. 59-71, 1990b.

Li, C. C., and A. Der Kiureghian, Optimal Discretization on random fields, Journal of Engineering Mechanics, Vol. 119, No. 6, pp. 1136-1154, 1993.

Mantoglou, A. M., and J. L. Wilson, The Turning Bands Method for Simulation of Random Fields Using Line Generation by a Spectral Method, *Water Resources Research*, Vol. 18, No. 5, pp. 1379-1394, 1982.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller, and M. Teller, Equations of State Calculations by Fast Computing Machines, *Journal of Chemical Physics*, Vol. 21, pp. 1087-1092, 1953.

Myer, L. R., A. M. Cook-Polek, L. J. Pyrak-Nolte, and C. Marone, Mercury porosimetry studies on a natural fracture, Proceedings, Fourth High-Level Radioactive Waste Management International Conference, Las Vegas, NV, 1993.

Narasimhan, T. N., and P. A. Witherspoon, An integrated finite difference method for analyzing fluid flow in porous media, *Water Resources Research*, Vol. 12, No. 1, pp. 57-64, 1976.

Neretnieks, I., T. Eriksen, and P. Tähtinen, Tracer movement in a single fissure in granitic rock: some experimental results and their interpretation, *Water Resou. Res.*, Vol. 18, No. 4, pp. 849-858, 1982.

Neretnieks, I., Solute transport in fractured rock-applications to radionuclide waste repositories, In Bear, J., Tsang, C. F., de Marsily, G. (Eds.), Flow and contaminant transport in fractured rock, Academic Press, San Diego, CA, pp. 39-127, 1993.

Nicholl, M. J., R. J. Glass, and S. W. Wheatcraft, Gravity-driven infiltration instability in initially dry nonhorizontal fractures, *Water Resources Research*, Vol. 30, No. 9, pp. 2533-2546, 1994.

Nolte, D. D., L. J. Pyrak-Nolte, and N. G. W. Cook, The fractal geometry of flow paths in natural fractures in rock and the approach to percolation, *Pure and Applied Geophysics*, Vol. 131, No. 1/2, pp. 111-138, 1989.

O'Sullivan, M. J., D. P. Bullivant, S. Follows, and W. I. Mannington, Modeling of the Wairakei-Tauhara geothermal system, Proceedings of the TOUGH Workshop 98', Ed., K. Pruess, Lawrence Berkeley National Laboratory Report, LBNL-41995, May, 1998.

Peaceman, D. W., *Fundamentals of numerical reservoir simulation*, Elsvier, Amsterdam, 176pp., 1977.

Persoff, P., and K. Pruess, Two-phase flow visualization and relative permeability measurement in natural rough-walled rock fractures, *Water Resou. Res.*, Vol. 31, No. 5, pp. 1175-1186, 1995.

Peters, R. R., and E. A. Klavetter, A continuum model for water movement in an unsaturated fractured rock mass, *Water Resources Research*, Vol. 24, No. 3, pp. 416-430, 1988.

Philip, J. R., Numerical solution of equations of the diffusion type with diffusivity concentration-dependent, *Transactions of the Faraday Society*, No. 391, pp. 885-982, 1955.

Philip, J. R., Theory of infiltration, *Adv. Hydrosci.*, 5, pp. 215-296, 1969.

Power, W. L., T. E. Tullis, S. R. Brown, G. N. Boitnott, and C. H. Scholz, Roughnes of natural fault surfaces, *Geophysical Research Letters*, Vol. 14, No. 1., pp. 29-32, 1987.

Press, W. H., S. A. Teukolsky, W. T., Vettering, and B. P. Flannery, Numerical Recipes in FORTRAN, The Art of Scientific Computing, 2nd ed., Cambridge University Press, New York, 963p., 1986.

Pruess , K., and E. Antunez, Applications of TOUGH2 to infiltration of Liquids in Media with Strong Heterogeneity, Proceedings of the TOUGH Workshop '95, Lawrence Berkeley Laboratory Report LBL-37200, pp. 69-76, Berkeley, CA, March, 1995.

Pruess, K., and Y. W. Tsang, On two-phase relative permeability and capillary pressure of rough-walled rock fractures, *Water Resources Research*, Vol. 26, No. 9, pp. 1915-1926, 1990.

Pruess, K., TOUGH User's Guide, Lawrence Berkeley National Laboratory report, LBL-20700, June 1987.

Pruess, K., On water seepage and fast preferential flow in heterogeneous, unsaturated rock fractures, *J. of Contaminant Hydrology,* Vol. 30, pp. 333-362, 1998.

Pruess, K., TOUGH2 - A General-Purpose Numerical Simulator for Multiphase Fluid and Heat Flow, Lawrence Berkeley National Laboratory report, LBNL-29400, May 1991.

Pruess, K., B. Faybishenko, and G. S. Bodvarsson, Alternative concepts and approaches for modeling flow and transport in thick unsaturated zones of fractured rocks, *J. of Contaminant Hydrology*, Vol. 38, pp. 281-322, 1999.

Pyrak-Nolte, L. J., L. R. Myer, N. G. W. Cook, and P. A. Witherspoon, Hydraulic and mechanical properties of natural fractures in low permeability rock, Proc. 6th international congress on rock mechanics (eds. Herget, G., and Vongpaisal, S.) Montreal, Canada (A. A. Balkema, Rotterdam), pp. 225-231,1987.

Pyrak-Nolte, L. J., N. G. W. Cook, and D. D. Nolte, Fluid percolation through single fractures, *Geophysical Research Letters,* Vol. 15, No. 11, p. 1247-1250, 1988.

Pyrak-Nolte, L. J., N. G. W. Cook, and L. R. Myer, A stratified percolation model for saturated and unsaturated flow through natural fractures, Proceedings of the Firth Annual International High-Level Radioactive Waste Management Conference, Las Vegas, NV, pp. 551-558, American Nuclear Society, La Grange Park, IL, 1990.

Raven, K. G., and J. E. Gale, Water flow in a natural rock fracture as a function of stress and sample size, *Int. J. Rock Mech. Min. Sci. & Geomech. Abstr.,* Vol. 22, No. 4, pp. 251-261, 1985.

Rice, J. A., Mathematical Statistics and Data Analysis, 2nd ed., 652pp., Duxbury Press, 1995.

Rubin, Y., Flow and transport in bimodal heterogeneous formations, *Water Resources Research*, Vol. 31, No. 10, pp. 2461-2468, 1995.

Sen, M. K., A., Datta-Gupta, P. L., Stoffa, L. W., Lake, and G. A., Pope, Stochastic Reservoir Modeling Using Simulated Annealing and Genetic Algorithm, *SPE Formation Evaluation*, Vol. 10, No. 1, pp. 49-55, 1995.

Sen, M. K., and P. L., Stoffa, Nonlinear One-Dimensional Seismic Waveform Inversion Using Simulated Annealing, *Geophysics*, Vol. 56, No. 10, pp. 1624-1638, 1991.

Senger, R., P. Marschall, and C. Bühler, Gas tracer transport in two-phase flow field: Numerical simulations and field experiments at the Grimsel Test Site (GTS), Switzerland, Proceedings of the TOUGH Workshop 98', Ed., K. Pruess, Lawrence Berkeley National Laboratory Report, LBNL-41995, May, 1998.

Shinozuka, M., and C. M. Jan, Digital simulation of random processes and its applications, *J. Sound and Vibrations*, Vol. 25, No. 1, pp. 111-128, 1972.

Thordarson, W., Perched groundwater in zeolitized-bedded tuff, Rainier Mesa and vicinity, Nevada Test Site, Nevada, US Geological Survey Report TEI-862, 1965.

Tompson, A. F. B., R. Ababou, and L. W. Gelhar, Implementation of the Three-Dimensional Turning Bands Random Field Generator, *Water Resources Research.*, Vol. 25, No. 10, pp. 227-2243, 1989.

Tsang, Y. W., and P. A. Witherspoon, Hydromechanical behavior of a deformable rock fracture subject to normal stress, *J. Geophy. Res.*, Vol. 86, pp. 9287-9298, 1981.

Tsang, Y. W., The effect of tortuosity on fluid flow through a single fracture, *Water Resou. Res.*, Vol. 20, pp. 1209-1215, 1984.

Walsh, J. B., Effect of pore pressure and confining pressure on fracture permeability, *Int. J. Rock Mech. Min. Sci. & Geomech. Abstr.*, Vol. 18, pp. 429-435, 1981.

Wang, J. S. Y., and T. N. Narasimhan, Hydrologic mechanisms governing fluid flow in a partially saturated, fractured, porous medium, *Water Resources Research*, Vol. 21, No. 12, pp. 1861-1874, 1985.

Wang, J. S. Y., and T. N. Narasimhan, Aperture correlation of a fractal fracture, *J. Geophys. Res.*, Vol. 93, No. B3, pp. 2216-2224, 1988.

Wang, J.S.Y., and T. N. Narasimhan, Unsaturated flow in fractured porous media, in: *Flow and Contaminant Transport in Fractured Rock*, Academic Press, 1993.

Webb, S. W., S. A. Finsterle, K. Pruess, and J. M. Phelan, Predication of the TNT signature from buried UXO/Landmines, Proceedings of the TOUGH Workshop 98', Ed., K. Pruess, Lawrence Berkeley National Laboratory Report, LBNL-41995, May, 1998.

Williams, S. A., and A. I. El-Kadi, COVAR: A Computer Program for Generating Two-Dimensional Fields of Autocorrelated Parameters by Matrix Decomposition,

Report, Int. Groundwater Model. Cent., Holcomb Res. Inst., Butler Univ., Indianapolis, Indiana, 1986.

Witherspoon, P. A., Wang, J. S. Y., K. Iwai, and J. E. Gale, Validity of cubic law for fluid flow in a deformable rock fracture, *Water Resou. Res.,* Vol. 16, No. 6, pp. 1016-1024, 1980.

Wood, T. R., and G. T. Norrell, Integrated large-scale aquifer pumping and infiltration tests, Groundwater Pathways, OU 7-06, Summary Report, INEEL-96/0256, Lockheed Martin Idaho Technologies, ID, 1996.

Xu, T., K, Pruess, and G. Brimhall, Introducing reactive chemical transport to TOUGH2: Application to supergene copper enrichment, Proceedings of the TOUGH Workshop 98', Ed., K. Pruess, Lawrence Berkeley National Laboratory Report, LBNL-41995, May, 1998.

Zimmerman, R. W., D. W. Chen, and N. G. W. Cook, The effect of contact area on the permeability of fractures, *J. of Hydrology,* Vol. 139, pp. 79-96, 1992.

Appendix A. Derivation of a semi-variogram model and its corresponding correlation function

(A) Definitions

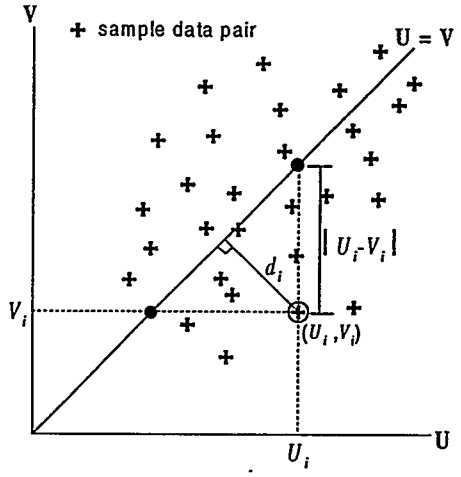


Figure A.1. Scatterplot of random variables **U** and **V**.

Dependence of two random variables **U** and **V** can be visualized from their scatterplot as shown in Figure A.1 (Isaaks and Srivastava, 1989), in which sample data of **U** and **V** measured at the same location are plotted against each other on the same diagram. Note that the solid line at 45° features the perfectly correlated pair, i.e., **U** = **V**. The spatial correlation of the two random variables is usually quantitatively measured by their covariance function, Cov[**U**, **V**]

$$\mathrm{Cov}[\mathbf{U},\mathbf{V}] = \mathrm{E}[(\mathbf{U} - m_{\mathbf{U}})(\mathbf{V} - m_{\mathbf{V}})] = \mathrm{E}[\mathbf{U}\cdot\mathbf{V}] - m_{\mathbf{U}} m_{\mathbf{V}} \tag{A-1}$$

where $m_{\mathbf{U}}$ and $m_{\mathbf{V}}$ are means of **U** and **V**, respectively. Covariance is used to measure the similarity between two random variables. However, the variability of two random variables is usually measured by their moment of inertia about the 45° line on their scatterplot. This quantity is called "semi-variogram" in the literature of geostatistics, which is written as

$$\gamma_{\mathbf{UV}} = \frac{1}{N}\sum_{i=1}^{N} d_i^2 = \frac{1}{2N}\sum_{i=1}^{N}(U_i - V_i)^2 \tag{A-2}$$

where N is the total number of random pairs (U_i, V_i), and d_i is distance on a scatterplot from a random pair (U_i, V_i) to the line on which **U** = **V**. Semi-variogram and covariance have the following relationship

$$\begin{aligned} 2\gamma_{\mathbf{UV}} &= \left[\frac{1}{N}\sum_{i=1}^{N} U_i^2 - m_{\mathbf{U}}^2\right] + \left[\frac{1}{N}\sum_{i=1}^{N} V_i^2 - m_{\mathbf{V}}^2\right] - \left[\frac{2}{N}\sum_{i=1}^{N} U_i V_i - 2m_{\mathbf{U}} m_{\mathbf{V}}\right] + (m_{\mathbf{U}} - m_{\mathbf{V}})^2 \\ &= s_{\mathbf{U}}^2 + s_{\mathbf{V}}^2 - 2\mathrm{Cov}[\mathbf{U},\mathbf{V}] + (m_{\mathbf{U}} - m_{\mathbf{V}})^2 \end{aligned} \tag{A-3}$$

where $s_{\mathbf{U}}$ and $s_{\mathbf{V}}$ are standard deviations of **U** and **V**, respectively. The above definitions of covariance and semi-variogram can be applied to two random variables measured at

the same location, e.g., U_i and V_i in Figure A.1; or the same attribute measured at different locations. The latter is usually used in the literature of geostatistics to measure the spatial variability of a spatially random function (SRF).

For a SRF **Z(u)**, its spatial variability is quantitatively measured by calculating the mean squared difference between a pair (**Z(u)**, **Z(u+h)**), where **h** is the separation distance between **Z(u)** and **Z(u+h)**. Based on Eq(A-2), the semi-variogram of **Z(u)** is

$$2\gamma_{\mathbf{Z}}(\mathbf{h}) = \frac{1}{N(h)} \sum_{i=1}^{N(h)} [Z(u_i) - Z(u_i + h)]^2 \equiv \mathrm{E}\left\{[\mathbf{Z(u)} - \mathbf{Z(u+h)}]^2\right\} \qquad \text{(A-4)}$$

If **Z** is a stationary SRF, i.e., the statistics of **Z** are independent of location, Eq(A-4) can be simplified as

$$2\gamma_{\mathbf{Z}}(\mathbf{h}) = \mathrm{E}\left\{[\mathbf{Z(u)} - \mathbf{Z(u+h)}]^2\right\} = 2\mathrm{Var}[\mathbf{Z(u)}] - 2\mathrm{Cov}[\mathbf{Z(u)}, \mathbf{Z(u+h)}] \qquad \text{(A-5)}$$

or

$$\gamma_{\mathbf{Z}}(\mathbf{h}) = \mathrm{C}_{\mathbf{Z}}(\mathbf{0}) - \mathrm{C}_{\mathbf{Z}}(\mathbf{h}) \qquad \text{(A-6)}$$

where $\mathrm{C}_{\mathbf{Z}}(\mathbf{0})$ is the variance of **Z**, and $\mathrm{C}_{\mathbf{Z}}(\mathbf{h})$ is the covariance of **Z** with separation distance **h**. Hereafter, the subscript in $\gamma_{\mathbf{Z}}(\mathbf{h})$ or $\mathrm{C}_{\mathbf{Z}}(\mathbf{h})$ will be omitted for simplicity if it is clear that what SRF is being dealt with.

(B) Mathematical models of γ(h) and C(h)

A function that can be used as a covariance function must be positive definite such that a function that is a weighted linear combination of n SRF's has a non-negative variance. From this property, other properties of C(**h**) can be inferred, which are (a) C(0)= Var(**Z(u)**) ≥ 0, (b) C(**h**) = C(-**h**), and (c) |C(**h**)| ≤ C(0). (Journel and Huijbregts, 1978).

In the literature of geostatistics, there are several models of semi-variogram that have shown to be positive definite. One of the most commonly used models is the exponential model which is defined as

$$\gamma(h) = s_{\mathbf{Z}}^2 \left[1 - \exp\left(-\frac{3h}{\xi}\right)\right] \qquad \text{(A-7)}$$

its corresponding covariance function is defined as

$$C(h) = s_{\mathbf{Z}}^2 \exp\left(-\frac{3h}{\xi}\right) \qquad \text{(A-8)}$$

where h = |**h**| is the magnitude of the separation distance, ξ is the range parameter, and $s_{\mathbf{Z}}^2$ is the sill (or the variance of the underlying random variable **Z**). Note that Eqs (A-7)

is an isotropic model, thus, $\gamma(\mathbf{h}) = \gamma(h)$. The "range" is defined as the distance at which the semi-variogram reaches 95% of the sill. That is, at $h = \xi$, equation (A-7) reduces to

$$\gamma(h) = s_Z^2[1-\exp(-3)] = 0.95 s_Z^2 \tag{A-9}$$

For any semi-variogram model, the integral scale λ, (or correlation length) of **Z** is defined as (Dagan, 1989)

$$\lambda = \frac{1}{s_Z^2}\int_0^\infty C(h)\,\mathrm{d}h \tag{A-10}.$$

Therefore, for an exponential semi-variogram, λ can be derived as

$$\lambda = \frac{1}{s_X^2}\int_0^\infty s_X^2 \cdot \exp\left(-\frac{3h}{\xi}\right)\mathrm{d}h = \frac{\xi}{3} \tag{A-11}$$

That is, for an exponential semi-variogram, the correlation length is one third of the range. See Figure A.2 for illustration.

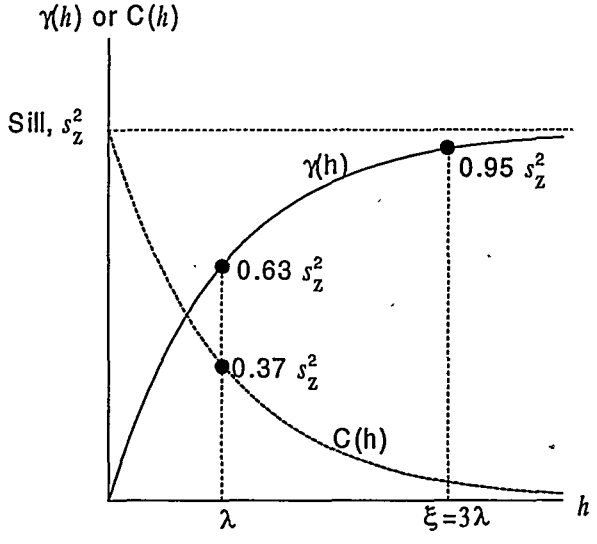


Figure A.2. Illustration of an exponential semi-variogram and its corresponding correlation function.

From Eq(A-11), Eq(A-7) can be recast as

$$\gamma(h) = s_Z^2\left[1-\exp\left(-\frac{h}{\lambda}\right)\right] \tag{A-12}$$

For an anisotropic exponential semi-variogram, Eq(A-12) can be rewritten as

$$\gamma(\mathbf{h}) = \gamma(h_x, h_z) = \mathrm{c} + s_Z^2 \left\{ 1 - \exp\left(-\sqrt{\left(\frac{h_x}{\lambda_x}\right)^2 + \left(\frac{h_z}{\lambda_z}\right)^2} \right) \right\} \qquad \text{(A-13)}$$

where λ_x and λ_z are the correlation lengths in the x and z directions, respectively; and h_x and h_z are the separation distances in the x and z directions, respectively. Note that this kind of anisotropy is called "geometric anisotropy" in the literature of geostatistics (Isaaks and Srivastava, 1989).

Appendix B. Nugget effect of a semi-variogram model

Theoretically, semi-variogram model at zero separation distance (h = 0) should have a zero value, i.e., $\gamma(0) = 0$, regardless of what type of model it may be. This property comes from that fact that the spatial correlation of **Z** to itself at $h = 0$ should be its univariate variance s_Z^2. When fitting a semi-variogram model to sample data, however, it may happen that the intercept extrapolated from the sample data to the vertical axis at the origin is a finite value but not zero. Likewise, the covariance function at large separation distance may approach asymptotically a finite value, not zero. This non-zero semi-variogram value at zero separation distance is called the "nugget effect" in the literature of geostatistics.

Nugget effect can be attributed to sampling error as well as small-scale variability (or microvariability) (Kitanidis, 1997). Usually, these two effects occur simultaneously in the field. Sampling error may be removed by following cautious sampling procedures, or using equipment with better precision, while taking field measurements. However, it is generally difficult to map the detailed variability of a spatially heterogeneous field by using finite sampling intervals. Thus, it is worthy to understand these effects and find an analytic way to describe them.

(A) Nugget effect due to sampling error

Denote the spatial random function by **Z**. At a particular location **u**, let the true value of **Z** be $z_0(\mathbf{u})$, and the measured value be $Z(\mathbf{u})$. Due to sampling error, $z_0(\mathbf{u})$ and $Z(\mathbf{u})$ may not be the same. Thus, we may write $Z(\mathbf{u})$ as the sum of $z_0(\mathbf{u})$, sampling error (ε) and a random fluctuation term ($\chi(\mathbf{u})$), i.e.,

$$Z(u) = z_0(\mathbf{u}) + \varepsilon + \chi(\mathbf{u}) \tag{B-1}$$

where $z_0(\mathbf{u})$ is a constant and ε is a constant random variable. Assume that ε has mean and variance as μ_ε and s_ε^2, respectively; and $\chi(\mathbf{u})$ is a random fluctuation term with mean and variance as 0 and s_χ^2, respectively. Furthermore, assume that $\chi(u)$ and $\chi(\mathbf{u}+\Delta\mathbf{u})$ are correlated to each other with a general correlation function $\psi_{\chi\chi}(\mathbf{u})$, and ε and ξ are independent to each other. Then, statistics of $\mathbf{Z}(\mathbf{u})$ can be derived as the followings

$$\begin{aligned} \mathrm{E}[\mathbf{Z}(\mathbf{u})] &= z_0(\mathbf{u}) + m_\varepsilon \\ \mathrm{Var}[\mathbf{Z}(\mathbf{u})] &= s_\varepsilon^2 + s_\chi^2 \\ \mathrm{Cov}[\mathbf{Z}(\mathbf{u}), \mathbf{Z}(\mathbf{u}+\Delta\mathbf{u})] &= s_\varepsilon^2 + \psi_{\chi\chi}(\mathbf{u}) s_\chi^2 = \mathrm{C}(\Delta\mathbf{u}) \equiv \mathrm{C}(h) \end{aligned} \tag{B-2}$$

Assuming positive correlation of χ, i.e., $0 \le \psi_{\chi\chi}(\mathbf{u}) \le 1$, then $s_\varepsilon^2 \le \mathrm{C}(h) \le s_\varepsilon^2 + s_\chi^2$. The correlation function $\mathrm{C}(h)$ is shown schematically in Figure B.1. It is obvious from Figure B.1 that the covariance function does not go to zero as h increases but go asymptotically to a constant s_ε^2. Statistically speaking, the contribution of s_ε^2 to the correlation function $\mathrm{C}(h)$ is called "bias" (Rice, 1995). In other words nugget effect may be resulted from a biased measurement.

(B) Nugget effect due to small-scale variability

To make measurements of a SRF in the field, it is inevitable to have a minimum sampling interval due to the constraint of cost and efficiency. That is, spatial variability within the minimum sampling interval may not be sampled. See Figure B.2(a) for illustration, in which the nugget effect is caused by the small-scale variability. In this case, however, there may still exist a semi-variogram that fits the sample data but has a zero value at $h = 0$, see Figure B.2(b). In other words, the semi-variogram at $h = 0$ may still be zero but jumps to a finite value at small separation distance due to small-scale variability.

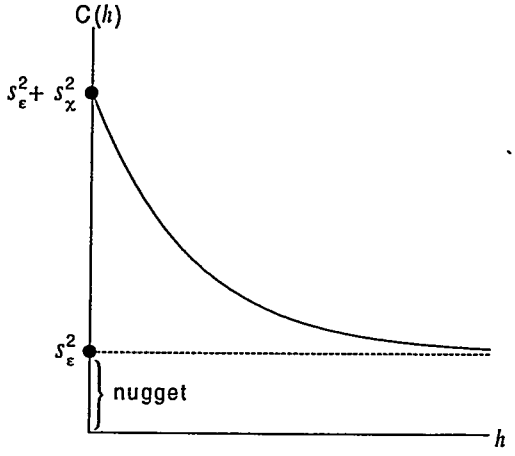


Figure B.1 Nugget effect due to sampling error (or bias).

To model the discontinuous jump at the origin, a semi-variogram model, e.g., exponential, with a nugget effect can be recast as

$$\gamma(h) = \begin{cases} 0 & \text{if } h = 0 \\ s_0^2 + s_Z^2\left[1 - \exp\left(-\dfrac{3h}{\xi}\right)\right] & \text{if } h > 0 \end{cases} \tag{B-3}$$

and the corresponding correlation function can be written as

$$C(h) = \begin{cases} s_0^2 + s_Z^2 & \text{if } h = 0 \\ s_Z^2 \exp\left(-\dfrac{3h}{\xi}\right) & \text{if } h > 0 \end{cases} \tag{B-4}$$

where s_0^2 is the nugget, and $s_0^2 + s_Z^2$ is the sill (Isaaks and Srivastava, 1989).

Equations (B-3) and (B-4) can be plotted in Figures B.3(a) and B.3(b), respectively. Figure B.3(a) shows that $\gamma(h)$ at the origin still has a zero value. As soon as h becomes larger than zero, there is a sudden jump from 0 to s_0^2. This sudden jump

reflects the fact that the sampling interval is too large to capture the spatial variability within the distance smaller than the sampling interval. The corresponding covariance function shows that C(h) at $h = 0$ is the summation of s_Z^2 and s_0^2 (the nugget). As soon as h becomes larger than 0, C(h) suddenly drops to s_Z^2. As h becomes even larger, C(h) decreases asymptotically to zero, suggesting as it should be because **Z**'s become uncorrelated to each other at large separation distances.

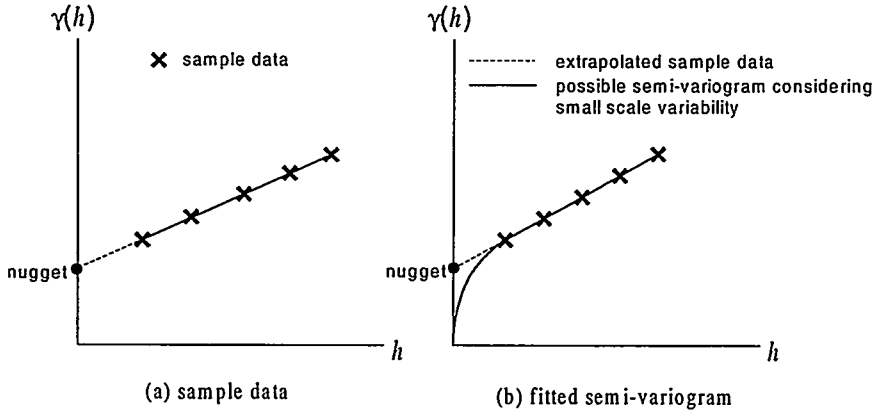


Figure B.2 Illustration of nugget effect due to small scale variability (from de Marsily, 1986).

One way to overcome the small-scale variability is to reduce the sampling interval. However, doing so may not be practical due to the extra number of sampling points. Thus, the other convenient alternative is to manipulate the definition of $\gamma(h)$ and introduce a discontinuity at the origin such as Eqs(B-3) and (B-4).

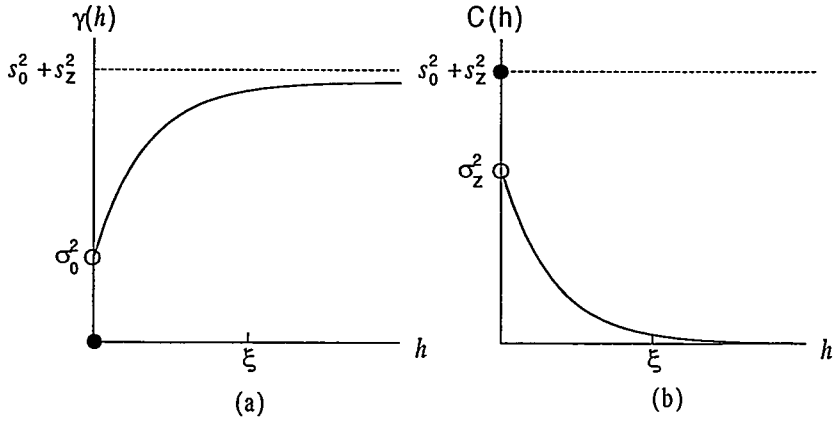
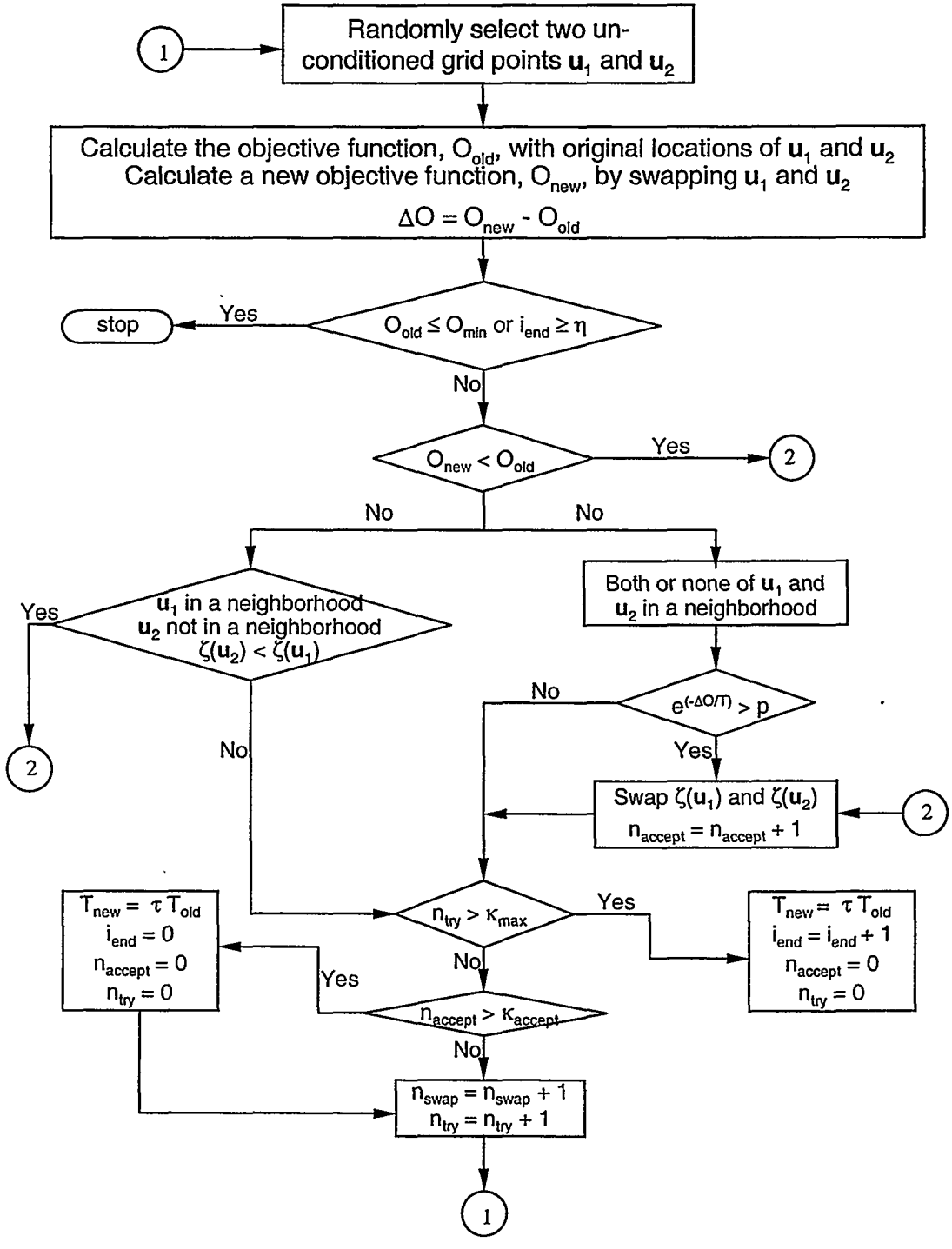


Figure B.3 Exponential semi-variogram and its corresponding correlation function with nugget effect due to small-scale variability.

Appendix C Source code of the modified Metropolis algorithm

(1) Flowchart


1
Randomly select two un-conditioned grid points u_1 and u_2
Calculate the objective function, O_{old}, with original locations of u_1 and u_2
Calculate a new objective function, O_{new}, by swapping u_1 and u_2
$\Delta O = O_{new} - O_{old}$
stop
Yes
$O_{old} \le O_{min}$ or $i_{end} \ge \eta$
No
$O_{new} < O_{old}$
Yes
2
No
No
Yes
u_1 in a neighborhood
u_2 not in a neighborhood
$\zeta(u_2) < \zeta(u_1)$
Both or none of u_1 and u_2 in a neighborhood
2
No
$e^{(-\Delta O/T)} > p$
No
Yes
Swap $\zeta(u_1)$ and $\zeta(u_2)$
$n_{accept} = n_{accept} + 1$
2
$T_{new} = \tau T_{old}$
$i_{end} = 0$
$n_{accept} = 0$
$n_{try} = 0$
$n_{try} > \kappa_{max}$
Yes
$T_{new} = \tau T_{old}$
$i_{end} = i_{end} + 1$
$n_{accept} = 0$
$n_{try} = 0$
No
Yes
$n_{accept} > \kappa_{accept}$
No
$n_{swap} = n_{swap} + 1$
$n_{try} = n_{try} + 1$
1

(2) Source code

```
C INCLUDE FILE 'METRO.INC'
```

```
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C                                                                          %
C Copyright (C) 1992 Stanford Center for Reservoir Forecasting. All rights reserved   %
C Distributed with: C.V. Deutsch and A.G. Journel.                         %
C ``GSLIB: Geostatistical Software Library and User's Guide," Oxford University Press, New York, 1992.   %
C                                                                          %
C The programs in GSLIB are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY. No author or   %
C distributor accepts responsibility to anyone for the consequences of using them or for whether they serve any particular purpose   %
C or work at all, unless he says so in writing. Everyone is granted permission to copy, modify and redistribute the programs in   %
C GSLIB, but only under the condition that this notice and the above copyright notice remain intact.   %
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c
c The following Parameters control static dimensioning within sasim3d:
c
c  MAXX     maximum nodes in X
c  MAXY     maximum nodes in Y
c  MAXZ     maximum nodes in Z
c  MAXCUT   maximum number of cutoffs/data to define CDF
c  MAXLAG   maximum number of lags in variogram calculation
c  MAXNST   maximum number of structures for variogram model
c
c Author: C.V. Deutsch                    Date: February 1990
c-----------------------------------------------------------------------------------------

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

      parameter (MAXX  = 100, MAXY  = 100, MAXZ  =  1,
     +           MAXCUT =5, MAXLAG = 500, MAXOBJ =  2,
     +           MAXNST =  4, EPSLON=1.0d-20,VERSION=1.200)
c
c Array declaration:
c
      real*8   cut(MAXCUT),cdf(MAXCUT),var(MAXX,MAXY,MAXZ),sas(6)
      real*8   varnew(MAXLAG,MAXOBJ),varmod(MAXLAG),
     +         varact(MAXLAG,MAXOBJ),sclfac(MAXLAG),
     +         vardiv(MAXLAG,MAXOBJ),ltpar,utpar,renorm
      integer  seed,part,report,ixl(MAXLAG),iyl(MAXLAG),izl(MAXLAG),
     +         it(MAXNST),utail,ltail
      logical  twopar,only2d,cond(MAXX,MAXY,MAXZ),comp
      real*8   cc(MAXNST),aa(MAXNST),ang1(MAXNST),ang2(MAXNST),
     +         ang3(MAXNST),anis1(MAXNST),anis2(MAXNST),
     +         gammanew(maxlag,2),gammah(maxlag,2),gammav(maxlag,2),
     +         divnew(maxlag,maxobj)
      real*8   gamunwt(maxlag)
c
c*************************************************************/
c 1/20/97 : The following lines are added by Tai-Sheng Liou
c
      real*8 ymean,ystd
      integer itrans,nbhd(maxx,maxy),defnbhd
      character datafl*40,outfl*40,dbgfl*40,condfl*40,lagfl*40,
     +        horvarfl*40,vervarfl*40,varfl*40,imageinfl*40,
     +        imageoutfl*40
c*************************************************************\
c Common blocks:
c
      common /grid3d/ xsiz,ysiz,zsiz,xmn,ymn,zmn,nx,ny,nz
      common /genral/ seed,nsim,var,sas,part,llag,limagein,limageout,
     +             lout,lvar,idbg,report,ldbg

      common /inimod/ cut,cdf,ltpar,utpar,ltail,utail,zmin,
     +             zmax,igauss,isill,ncdf
```

```
      common /variog/ sclfac,varnew,varact,vardiv,
     +              varmod,divnew,renorm,nlag,
     +              neighbor,ixl,iyl,izl
      common/variog1/ gamunwt
      common /cova3d/ c0,cmax,cc,aa,ang1,ang2,ang3,anis1,anis2,nst,it
      common /logics/ twopar,only2d,cond
      common /compwt/ wtcomp,oinit,pmsum,pmnew,gamdf,comp,nwt
c
C************************************************************/
c 1/20/97 : The following common blocks are added by Tai-Sheng Liou
c
      common /lognorm/ ymean,ystd,pcut,itrans
      common /weight/ wfedge,wfcond,scale,iedge,icond
      common /anisop/ lhvar,lvvar,noisop
      common /datapt/ noncond
      common /cutoff/ xcut0,aspcut,xcut,ptarget,cutsave,paspsave
      common /fname/  datafl,outfl,dbgfl,condfl,horvarfl,vervarfl,
     + varfl,lagfl,imageinfl,imageoutfl,intervar,nswap
      common /gamma/ gammanew,gammah,gammav
      common /neighbor/ nbhd
      common/annealing/metro,defnbhd,imod,ifield
C***********************************************************\
```

C MAIN PROGRAM 'SASIM' (the calling program for SA)

```
         subroutine sasim
c-------------------------------------------------------------------------------------------------------------------------------------------

C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C                                                                                                    %
C Copyright (C) 1992 Stanford Center for Reservoir Forecasting.  All rights reserved.               %
C Distributed with: C.V. Deutsch and A.G. Journel                                                  %
C ``GSLIB: Geostatistical Software Library and User's Guide," Oxford University Press, New York, 1992 %
C                                                                                                    %
C The programs in GSLIB are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY.  No author or %
C distributor accepts responsibility to anyone for the consequences of using them or for whether they serve any particular purpose %
C or work at all, unless he says so in writing.  Everyone is granted permission to copy, modify and redistribute the programs in %
C GSLIB, but only under the condition that this notice and the above copyright notice remain intact. %
C                                                                                                    %
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c-----------------------------------------------------------------------
c             3-D Simulation by Annealing
c-----------------------------------------------------------------------
c
c Conditionally Simulate a Complete 3-D Field with Simulated Anealing.
c
c The objective function is the squared difference between the desired variogram and the actual variogram for as many lags as
c specified. The objective function may be in two parts - one part includes the conditioning data and the other includes pairs of
c simulated data only.

c INPUT/OUTPUT Parameters:
c
c  - Name of a data file of conditioning data (GEOEAS format)
c  - column numbers for x, y, z, and variable
c  - trimming limits (used to flag missing values)
c  - flag specifying whether a standard Normal deviate is to be simulated (set to 1)
c  - Name of a data file for non-parametric distribution
c  - column numbers for variable and weight
c  - data limits (used for tail extrapolation)
c  - option and parameter for the lower tail
c  - option and parameter for the upper tail
c  - An output file (may be overwritten)
c  - A output file for variograms (may be overwritten)
c  - The debugging level (integer code - larger means more)
```

```
c  - A file for the debugging output
c  - Whether or not to use an automatic annealing schedule (0=auto)
c  - annealing schedule
c  - Whether a one part or a two part objective function is used
c  - Random Number Seed
c  - The number of simulations
c  - X grid definition (number, minimum, size): nx,xmn,xsiz
c  - Y grid definition (number, minimum, size): ny,ymn,ysiz
c  - Z grid definition (number, minimum, size): nz,zmn,zsiz
c  - The maximum number of lags to be considered
c  - Variogram Definition:  number of structures(nst), nugget, and whether or not to renormalize sill to the variance(0=auto)
c  - the next "nst*2" lines require:
c     First line:
c       a) an integer code for variogram (1=sph,2=exp,3=gaus,4=pow)
c       b) "a" parameter (range except for power model)
c       b) "c" parameter (contribution except for power model).
c     Second line:
c       a) azimuth principal direction (measured clockwise from Y).
c       b) dip of principal direction (measured negative down from X).
c       c) a third rotation of the two minor directions about the principal direction. This angle acts counterclocwise
c          when looking in the principal direction.
c       Two anisotropy factors are required to complete the definition
c       of the geometric anisotropy of each nested structure:
c       d) radius in minor direction at 90 degrees from the principal direction divided by the principal radius.
c       e) radius in minor direction at 90 degrees vertical from the principal direction divided by the principal radius.
c
c The output file will be a GEOEAS file containing the simulated values The file is ordered by x,y,z, and then simulation (i.e., x cycles
c fastest, then y, then z, then simulation number).
c
c Original: C.V. Deutsch                    Date: April 1990
c
c Definitions of some variables
c
c  varact(il,io)      - sum of squared difference between the weighted variogram and the model variogram
c  varnew(il,io)      - same as 'varact' but used in the subroutine 'OBJECT'. If perturbation accepted, set varact=varnew; otherwise,reset
c                       varact to the previous value at the next perturbation
c  var_unwt(il,io)    - sum of squared difference between the un-weighted variogram and the model variogram. This variogram value is
c                       used to test the convergence
c  varw1(il,io)       - same as var_unwt but used in the subroutine 'OBJECT' If perturbation accepted, set var_unwt=varw1; otherwise,
c                       reset var_unwt to the previous value at the next perturbation
c  vardiv(il,io)      - number of pairs at il-th lag for weighted variogram
c  divnew(il,io)      - same as vardiv but used in 'OBJECT'. It will be restored to the previous value if a perturbation is rejected.
c  div_unwt(il,io)    - number of pairs at il-th lag for unweighted variogram
c-----------------------------------------------------------------------------------------------------------------------------------

C************************************************************/
c
c  Updated by Tai-Sheng Liou, 4/22/1997
c
include 'metro.inc'
    logical  accept,first,vgmout(6)
    real*8 actsv(30,2),divsv(30,2)
c
c Read the data (Initialize) and find the starting objective function:
c
    open(50,file='obj.dat',status='unknown')
    first=.true.
    do i=1,6
      vgmout(i)=.false.
    end do
    do 10 i=1,nx
    do 10 j=1,ny
      nbhd(i,j)=0
10   continue

c
c 6/28/97 : Find the neighborhood of aspersity contacts
c nbhd=1, neighborhood is defined as a square
```

```
c nbhd=2, neighborhood is defined such that the distance between a grid block and an asperity contact is smaller  than the size of
c           neighborhood
c
c
      open(777,file='neighbor.dat',status='unknown')
      size=real(neighbor)
      do 12 i=1,nx
      do 12 j=1,ny
      do 12 k=1,nz
        if (cond(i,j,k)) then
          do ii=-neighbor,neighbor
            do jj=-neighbor,neighbor
            ix=i+ii
            iy=j+jj
            iz=k
            if (defnbhd.eq.2) then
              dx=real(ii)
              dy=real(jj)
              dist=sqrt(dx*dx+dy*dy)
            end if
            if (ix.ge.1.and.ix.le.nx.and.iy.ge.1.and.iy.le.ny.and.
     +          iz.ge.1.and.iz.le.nz) then
              if (defnbhd.eq.1) then
                if(.not. cond(ix,iy,iz).and.nbhd(ix,iy).eq.0)then
                  nbhd(ix,iy)=1
                  xasp=xmn+real(ix-1)*xsiz
                  yasp=ymn+real(iy-1)*ysiz
                  write(777,778) xasp,yasp,0,wfcond
                end if
              elseif(defnbhd.eq.2) then
                if(.not. cond(ix,iy,iz).and.nbhd(ix,iy).eq.0.
     +             and.dist.le.size) then
                nbhd(ix,iy)=1
                xasp=xmn+real(ix-1)*xsiz
                yasp=ymn+real(iy-1)*ysiz
                write(777,778) xasp,yasp,0,wfcond
              end if
            end if
            end if
            end do
          end do
        end if
12    continue
778   format(2(f6.1,1x),i4,2x,f5.1)
      close(777)
C*************************************************************\
c
c 1/28/97 : Write header to lagfl if the filename of lagfl is not 'nodata.dat'
c
c     if (first.and.lagfl(1:10).ne.'nodata.dat') then
c       write(llag,9990)
c     end if

      call initob(obj,first)
      first=.false.
c
c Initial Conditions:
c
      nswap  = 0
      iend   = 0
      temp   = sas(1)
      accept = .false.
      if(only2d) then
          k1 = 1
          k2 = 1
      endif
c
c Loop until convergence or the stopping number:
```

```
c
 1   naccept = 0
     ntry    = 0
     write(ldbg,*) '  Obj. Fun.   # of swap'
     if(idbg.gt.2) then
         write(*,777) obj
         write(*,996) temp,nswap
         write(ldbg,996) temp,nswap
         write(ldbg,997) obj,nswap
         write(50,*) nswap,obj
 996       format(' New Temperature:    ',e12.5,' Total swaps: ',i12)
 997       format('0 ',e14.7,1x,i12)
 777       format(' Objective function: ',e14.7)
     endif
c
c Keep attempting to swap values until some limit is exceeded:
c
 2   ntry = ntry + 1
     nswap = nswap + 1
     if(idbg.gt.2) then
         if((int(nswap/report)*report).eq.nswap) then
           write(*,998) obj,nswap
           write(ldbg,999) obj,nswap
           write(50,*) nswap,obj
         endif
 998       format(' Objective Function: ',e14.7,' Total swaps: ',i12)
 999       format('1 ',e14.7,1x,i12)
     endif
c
c Find a random pair such that none of the data is a conditioning point
c
 3   i1 = int(getrand(seed)*nx)+1
     j1 = int(getrand(seed)*ny)+1
     if(.not.only2d)
    +k1 = int(getrand(seed)*nz)+1
     if(cond(i1,j1,k1)) go to 3
 4   i2 = int(getrand(seed)*nx)+1
     j2 = int(getrand(seed)*ny)+1
     if(.not.only2d)
    +k2 = int(getrand(seed)*nz)+1
     if(cond(i2,j2,k2)) go to 4
     if(i2.eq.i1.and.j2.eq.j1.and.k2.eq.k1) go to 4

c
c Calculate Objective Function:
c
      call object(i1,j1,k1,i2,j2,k2,accept,objtry)
c
c Accept the swap if the objective has gone down and with a certain probability if the objective has gone up:
c
     accept = .false.
     if(objtry.gt.obj) then
       unif = dmax1(EPSLON,getrand(seed))
       if(metro.eq.1) then
         if(imod.eq.1) then


C*************************************************************/
C
c-------------------------------------------------------------------------
c  MOD1:
c      both P1 and P2 in nbhd     : standard Metropolis
c      none of P1 and P2 in nbhd. : standard Metropolis
c      P1 in nbhd but not P2      : standard Metropolis and P2<P1
c-------------------------------------------------------------------------
           if(nbhd(i1,j1).eq.1.and.nbhd(i2,j2).eq.1) then
             if(objtry.lt.(obj-temp*dlog(unif))) accept = .true.
           elseif(nbhd(i1,j1).ne.1.and.nbhd(i2,j2).ne.1) then
```

```
            if(objtry.lt.(obj-temp*dlog(unif))) accept = .true.
          elseif(nbhd(i1,j1).eq.1.and.nbhd(i2,j2).ne.1) then
            if(objtry.lt.(obj-temp*dlog(unif)).and.
     +        var(i2,j2,k2).lt.var(i1,j1,k1))accept=.true.
          else
            if(objtry.lt.(obj-temp*dlog(unif)).and.
     +        var(i1,j1,k1).lt.var(i2,j2,k2))accept=.true.
          end if
        elseif(imod.eq.2) then
c-------------------------------------------------------------------------
c MOD2:
c     both P1 and P2 in nbhd    : reject
c     none of P1 and P2 in nbhd. : standard Metropolis
c     P1 in nbhd but P2 not     : standard Metropolis and P2<P1
c-------------------------------------------------------------------------
          if(nbhd(i1,j1).eq.1.and.nbhd(i2,j2).eq.1) then
            accept=.false.
          elseif(nbhd(i1,j1).ne.1.and.nbhd(i2,j2).ne.1) then
            if(objtry.lt.(obj-temp*dlog(unif))) accept = .true.
          elseif(nbhd(i1,j1).eq.1.and.nbhd(i2,j2).ne.1) then
            if(objtry.lt.(obj-temp*dlog(unif)).and.
     +        var(i2,j2,k2).lt.var(i1,j1,k1))accept=.true.
          else
            if(objtry.lt.(obj-temp*dlog(unif)).and.
     +        var(i1,j1,k1).lt.var(i2,j2,k2))accept=.true.
          end if
        elseif (imod.eq.3) then
c-----------------------------------------------------------------
c MOD3:
c     both P1 and P2 in nbhd    : standard Metropolis
c     none of P1 and P2 in nbhd. : standard Metropolis
c     P1 in nbhd but P2 not     : P2<P1
c-----------------------------------------------------------------
          if(nbhd(i1,j1).eq.1.and.nbhd(i2,j2).eq.1) then
            if(objtry.lt.(obj-temp*dlog(unif))) accept = .true.
          elseif(nbhd(i1,j1).ne.1.and.nbhd(i2,j2).ne.1) then
            if(objtry.lt.(obj-temp*dlog(unif))) accept = .true.
          elseif(nbhd(i1,j1).eq.1.and.nbhd(i2,j2).ne.1) then
            if(var(i2,j2,k2).lt.var(i1,j1,k1))accept=.true.
          else
            if(var(i1,j1,k1).lt.var(i2,j2,k2))accept=.true.
          end if
        elseif(imod.eq.4) then
c-----------------------------------------------------------------
c MOD4:
c     both P1 and P2 in nbhd    : reject
c     none of P1 and P2 in nbhd. : standard Metropolis
c     P1 in nbhd but P2 not     : P2<P1
c-----------------------------------------------------------------
          if(nbhd(i1,j1).eq.1.and.nbhd(i2,j2).eq.1) then
            accept = .false.
          elseif(nbhd(i1,j1).ne.1.and.nbhd(i2,j2).ne.1) then
            if(objtry.lt.(obj-temp*dlog(unif))) accept = .true.
          elseif(nbhd(i1,j1).eq.1.and.nbhd(i2,j2).ne.1) then
            if(var(i2,j2,k2).lt.var(i1,j1,k1))accept=.true.
          else
            if(var(i1,j1,k1).lt.var(i2,j2,k2))accept=.true.
          end if
        end if
      else
c-------------------------------------------------------------------------
c Standard Metropolis considering the acceptance probability
c-------------------------------------------------------------------------
        if(objtry.lt.(obj-temp*dlog(unif))) accept = .true.
      end if
    else
        accept = .true.
    endif
```

```
      if(ifield.eq.1) then
        if(nswap.eq.1e4)vgmout(1)=.true.
        if(nswap.eq.1e5)vgmout(2)=.true.
        if(nswap.eq.1e6)vgmout(3)=.true.
        if(nswap.eq.1e7)vgmout(4)=.true.
        if(nswap.eq.2.e7)vgmout(5)=.true.
        if(nswap.eq.3.e7)vgmout(6)=.true.
      end if

c
c If we are keeping it then update the variogram arrays:
c
      if(accept) then
        n1=1e4
        if (ifield.eq.1) then
        do kk=1,6
          if(kk.le.4)n2=n1*10
          if(kk.ge.5)n2=n1+1e7
          if(vgmout(kk))then
           if(nswap.ge.n1.and.nswap.lt.n2) then
            vgmout(kk)=.true.
            no=1230+kk
            no1=1250+kk
            if(kk.eq.1)open(no,file='gamma4.dat',status='unknown')
            if(kk.eq.2)open(no,file='gamma5.dat',status='unknown')
            if(kk.eq.3)open(no,file='gamma6.dat',status='unknown')
            if(kk.eq.4)open(no,file='gamma7.dat',status='unknown')
            if(kk.eq.5)open(no,file='gamma71.dat',status='unknown')
            if(kk.eq.6)open(no,file='gamma72.dat',status='unknown')
            if(kk.eq.1)open(no1,file='real4.dat',status='unknown')
            if(kk.eq.2)open(no1,file='real5.dat',status='unknown')
            if(kk.eq.3)open(no1,file='real6.dat',status='unknown')
            if(kk.eq.4)open(no1,file='real7.dat',status='unknown')
            if(kk.eq.5)open(no1,file='real71.dat',status='unknown')
            if(kk.eq.6)open(no1,file='real72.dat',status='unknown')
           end if
          end if
          if(kk.le.3)n1=n1*10
          if(kk.ge.4)n1=n1+1e7
        end do
        end if
        do 5 ilag=1,nlag
        do 5 iobj=1,part
            varact(ilag,iobj) = varnew(ilag,iobj)
            vardiv(ilag,iobj) = divnew(ilag,iobj)
            actsv(ilag,iobj)  = varnew(ilag,iobj)
            divsv(ilag,iobj)  = divnew(ilag,iobj)
5       continue
        naccept      = naccept + 1
        obj          = objtry
        vartemp      = var(i1,j1,k1)
        var(i1,j1,k1) = var(i2,j2,k2)
        var(i2,j2,k2) = vartemp
        if(ifield.eq.1) then
        do kk=1,6
          no=1230+kk
          no1=1250+kk
          if (vgmout(kk)) then
            do 11 j=1,nlag
              do 11 k=1,part
                varact(j,k) = 0.0d0
                vardiv(j,k) = 0.0d0
11          continue
C*************************************************************\
c
c Calculate the Experimental Variogram:
c
      do 31 ix=1,nx
```

```
      do 31 iy=1,ny
      do 31 iz=1,nz
          v1 = var(ix,iy,iz)
          do 41 il=1,nlag
              iix  = ix + ixl(il)
              jjy  = iy + iyl(il)
              kkz  = iz + izl(il)
              if(iix.ge.1.and.iix.le.nx.and.
     +           jjy.ge.1.and.jjy.le.ny.and.
     +           kkz.ge.1.and.kkz.le.nz) then
                v2 = var(iix,jjy,kkz)
                io = 1
                varact(il,io) = varact(il,io)+
     +           (v1-v2)*(v1-v2)
                vardiv(il,io) = vardiv(il,io)+2.d0
              endif
 41       continue
 31   continue
          write(no,*) ' nswap = ',nswap
          write(no,500)
          do il=1,nlag
            dx = dble(ixl(il)) * xsiz
            dy = dble(iyl(il)) * ysiz
            dz = dble(izl(il)) * zsiz
            dx = sqrt(dx*dx+dy*dy+dz*dz)
            write(no,105) il,dx,varmod(il),
     +       varact(il,1)/vardiv(il,1),actsv(il,1)/divsv(il,1)
          end do
          do 13 iz=1,nz
          do 13 iy=1,ny
          do 13 ix=1,nx
              xx=xmn+xsiz*dble(ix-1)
              yy=ymn+ysiz*dble(iy-1)
              zz=zmn+zsiz*dble(iz-1)
              if(cond(ix,iy,iz)) then
                if (var(ix,iy,iz).eq.xcut0)
     +              var(ix,iy,iz)=0.0d0
                write(no1,701) xx,yy,zz,var(ix,iy,iz)
              else
                write(no1,702) xx,yy,zz,var(ix,iy,iz)
              endif
 13       continue
 701      format(3(f8.2,2x),f12.4,'  c')
 702      format(3(f8.2,2x),f12.4)
 105      format(i4,f10.4,6f18.8)
          vgmout(kk)=.false.
          close(no)
          close(no1)
          end if
        end do
        end if
        end if
c
c Converged to a Solution?
c
C*******************************************************************************************/
c
c 1/21/97 : The following lines are added to show the users why the program is terminated , Tai-Sheng Liou
c
c Test the convergence based on the un-weighted variogram: var_unwt
c
      if(obj.le.sas(6).or.iend.ge.sas(5)) then
          write(50,*) nswap,obj
          if (intervar.gt.nswap) write(*,*)
     +      'intervar = ',intervar,' > ',nswap
          if (obj.le.sas(6)) then
            write(*,600) nswap,obj,sas(6)
            write(ldbg,600) nswap,obj,sas(6)
```

```
        end if
        if (iend.ge.sas(5)) then
          write(*,*) 'Have tried ',iend,
     $          sas(3),' iterations -Program terminated'
          write(ldbg,*) 'Have tried ',iend,
     $          sas(3),' iterations -Program terminated'
        end if
        if (part.eq.1) then
          if(.not.first) write(lvar,500)
          if (noisop.eq.1 .and. .not.first) then
            write(lhvar,500)
            write(lvvar,500)
          end if
        else
          if (.not.first) write(lvar,510)
          if (noisop.eq.1 .and. .not.first) then
            write(lhvar,510)
            write(lvvar,510)
          end if
        end if
C*******************************************************************************\
c
c Calculate the experimental semi-variogram of the final image as a final check of the simulation
c
        first=.false.
        call initob(obj,first)
        rms=0.0d0
        do il=1,nlag
          dx = dble(ixl(il)) * xsiz
          dy = dble(iyl(il)) * ysiz
          dz = dble(izl(il)) * zsiz
          dx = sqrt(dx*dx+dy*dy+dz*dz)
          write(lvar,102) il,dx,varmod(il),
     +     gammanew(il,1),gammanew(il,2),gamunwt(il)
          if (noisop.eq.1) then
            if (noisop.eq.1) then
              if(iyl(il).eq.0) write(lhvar,102) il,dx,
     $          varmod(il),gammah(il,1),gammah(il,2)
              if(ixl(il).eq.0) write(lvvar,102) il,dx,
     $          varmod(il),gammav(il,1),gammav(il,2)
            end if
          end if
          rms=rms+(varmod(il)-gamunwt(il))**2.d0
        end do
        rms=dsqrt(rms)
        write(lvar,*)
        write(lvar,505) rms
505     format('RMS of semi-variogram = ',f9.4)
        return
      endif
c
c Tried too many at this "temperature"?
c
      if(ntry.gt.sas(3)) then
        iend = iend + 1
        temp = sas(2) * temp
        go to 1
      endif
c
c Accepted enough at this "temperature"?
c
      if(naccept.gt.sas(4)) then
        temp = sas(2) * temp
        iend = 0
        go to 1
      endif
c
c Go back for another attempted swap:
```

```
c
102  format(i4,f10.4,6f12.4)
500  format(3x,'i',4x,'h',9x,'r(model)',4x,'r(actual)',15x,'r(final)')
510  format(3x,'i',4x,'h',9x,'r(model)',4x,'r(simu.)',17x,'r(cond.)')
600  format(' At ',i12,' -th iter., obj = ',e12.5,' <',e12.5)
9990 format(' lag',2x,' t2   w2  ',' V1-',6x,'V1',7x,'t1   w1 ',
     +      '  V1+',4x,'t4   w4',5x,'V2-',6x,'V2',7x,'t3   w3  ',
     +      2x,'V2+',7x,'r(h)',10x,'r_exp(h,1)',6x,
     +      'r_exp(h,2)')
     go to 2
     end
c
c-----------------------------------------------------------------------------------------------------------
c Little function to shorten the calling arguments each time a random number is needed
c-----------------------------------------------------------------------------------------------------------
      real*8 function getrand(seed)
      implicit real*8(a-h,o-z)
      real*8   randnu(1)
      integer seed
      call rand(seed,1,randnu)
      getrand = randnu(1)
      return
      end
c

      subroutine initob(obj,first)
c-----------------------------------------------------------------------------------------------------------
c              Routines to Compute Objective Function
c          ************************************
c
c The objective function is the squared difference from the model variogram and the experimental variogram.
C The user specifies the lag separation distances and the number of lags that contribute to the objective function.
c
c 1. Initial Objective Function - Compute Both the Experimental and the Model Variograms. Compute the objective function as the
c    squared difference between the actual and the model variograms:
c 2. The second routine updates the variogram when a swap is being considered.
c
c Author: C.V. Deutsch                    Date: April 1990
c-----------------------------------------------------------------------------------------------------------
     include  'metro.inc'
     logical  first,image
c
c Initialize the varigoram arrays:
c
     do 1 j=1,nlag
     if(.not. first) gamunwt(j)=0.d0
     do 1 k=1,part
          varact(j,k) = 0.0d0
          vardiv(j,k) = 0.0d0
1    continue
c
c Calculate the Experimental Variogram:
c
     do 3 ix=1,nx
     do 3 iy=1,ny
     do 3 iz=1,nz
c
c Consider the first value in the pair and all directions and lags:
c
          v1 = var(ix,iy,iz)
          do 4 il=1,nlag
               ii  = ix + ixl(il)
               jj  = iy + iyl(il)
               kk  = iz + izl(il)
               if(ii.ge.1.and.ii.le.nx.and.
     +            jj.ge.1.and.jj.le.ny.and.
     +            kk.ge.1.and.kk.le.nz) then
c
```

```
c Found a pair that should go in calculation. Possibly keep the pairs involving a conditioning data separately:
c
                v2 = var(ii,jj,kk)
                io = 1
                varact(il,io) = varact(il,io)+
     +           (v1-v2)*(v1-v2)
                vardiv(il,io) = vardiv(il,io)+2.d0
              endif
 4          continue
 3        continue

c
c Reture if obj=-1, i.e., obj=-1 for checking the experimental variogram at certain iterations
c
      if(obj.eq.-1.d0) return
c
c write out the experimental varigoram:
c

      obj = 0.0d0
      do 5 il=1,nlag
        if (first) then
          dx = dble(ixl(il)) * xsiz
          dy = dble(iyl(il)) * ysiz
          dz = dble(izl(il)) * zsiz
          dx       = sqrt(dx*dx+dy*dy+dz*dz)
          demon=varmod(il)
          sclfac(il) = 1.0d0/(dmax1(1.d-4,demon*demon))
        end if
c
c Compute the objective function while we're at it:
c
          if(vardiv(il,1).le.0.0d0) then
              write(*,*) 'ERROR: lag ',il
              write(*,*) 'there are no pairs!!'
              stop
          endif
          if(part.eq.1) then
              act = varact(il,1)/vardiv(il,1)
              obj = obj + (varmod(il)-act)
     +                 * (varmod(il)-act)
     +                 * sclfac(il)
              if(.not. first) gamunwt(il)=act
          else
              act1 = varact(il,1)/vardiv(il,1)
              obj  = obj + (varmod(il)-act1)
     +                 * (varmod(il)-act1)
     +                 * sclfac(il)
              if(vardiv(il,io).gt.0.5d0) then
                act2 = varact(il,2)/vardiv(il,2)
                obj  = obj + (varmod(il)-act2)
     +                   * (varmod(il)-act2)
     +                   * sclfac(il)
              endif
          endif
 5      continue

c
c Normalize the scale factors so that the initial objective function is 1.0:
c
      if(first) then
          inquire(file=imageinfl,exist=image)
          if(image) then
            renorm=1.d0
          else
            renorm = 1.0d0 / obj
          end if
```

```
        do 9 il=1,nlag
            sclfac(il) = sclfac(il) * renorm
9       continue
        obj  = 1.0d0
      endif
      if(first) then
        open(8888,file='initvar.dat',status='unknown')
        do il=1,nlag
          dx=ixl(il)*xsiz
          dy=iyl(il)*ysiz
          dz=izl(il)*zsiz
          h=sqrt(dx*dx+dy*dy+dz*dz)
          if(io.eq.1) write(8888,*) h,varmod(il),
     +      varact(il,1)/vardiv(il,1)
          if(io.eq.2) write(8888,*) h,varmod(il),
     +      varact(il,1)/vardiv(il,1),varact(il,2)/vardiv(il,2)
        end do
        close(8888)
      end if

102   format(i4,f10.4,3f12.4)
500   format('  i       h    r(model)   r(actual)')
510   format('  i       h    r(model)   r(simu.)  r(cond.)')
c
c Return with the current objective function:
c
      return
      end
c-----------------------------------------------------------------------------------------------------------------------------------
c
c Considering a swap: Update the Experimental Variogram and then compute the objective function as the squared difference between
c the actual and the model variogram.
c
c Author: C.V. Deutsch                    Date: April 1990
c-----------------------------------------------------------------------------------------------------------------------------------
      subroutine object(i1,j1,k1,i2,j2,k2,accept,objnew)
      include  'metro.inc'
      logical  accept

c
c Ensure that the experimental variogram array values are current. If the last swap was accepted then we don't have to update the new
c array, otherwise we have to reset back to the correct variogramarray:
c
      if (nswap.le.50.and.lagfl(1:10).ne.'nodata.dat')
     +  write(llag,*) nswap,' -th perturbation'
      if (nswap.le.50.and.lagfl(1:10).ne.'nodata.dat')
     +  write(llag,*)
      if(.not.accept) then
          do 10 il=1,nlag
          do 10 io=1,part
              varnew(il,io) = varact(il,io)
              divnew(il,io) = vardiv(il,io)
10        continue
      endif
      v1 = var(i1,j1,k1)
      v2 = var(i2,j2,k2)

c
c MAIN LOOP to consider the change to all lags and directions:
c
      do 20 il=1,nlag

c-------------------------------------------------------------------
c Update the variogram near the first point (positive lag):
c-------------------------------------------------------------------
          ii = i1 + ixl(il)
          jj = j1 + iyl(il)
          kk = k1 + izl(il)
```

```
            if(ii.ge.1.and.ii.le.nx.and.
     +         jj.ge.1.and.jj.le.ny.and.
     +         kk.ge.1.and.kk.le.nz) then
              if(ii.ne.i2.or.jj.ne.j2.or.kk.ne.k2) then
                v0 = var(ii,jj,kk)
                io=1
                if (twopar) then
                 if(nbhd(i1,j1).eq.1.and.cond(ii,jj,kk))io=2
                 if(cond(i1,j1,k1).and.nbhd(ii,jj).eq.1)io=2
                 if(nbhd(i1,j1).eq.1.and.nbhd(ii,jj).eq.1)io=2
                end if
                varnew(il,io) = varnew(il,io)
     +          -(v1-v0)*(v1-v0)+(v2-v0)*(v2-v0)
              end if
            end if

c-----------------------------------------------------------------------
c     Update the variogram near the first point (negative lag):
c-----------------------------------------------------------------------
            ii = i1 - ixl(il)
            jj = j1 - iyl(il)
            kk = k1 - izl(il)
            if(ii.ge.1.and.ii.le.nx.and.
     +         jj.ge.1.and.jj.le.ny.and.
     +         kk.ge.1.and.kk.le.nz) then
              if(ii.ne.i2.or.jj.ne.j2.or.kk.ne.k2) then
                v0 = var(ii,jj,kk)
                io=1
                if (twopar) then
                 if(nbhd(i1,j1).eq.1.and.cond(ii,jj,kk))io=2
                 if(cond(i1,j1,k1).and.nbhd(ii,jj).eq.1)io=2
                 if(nbhd(i1,j1).eq.1.and.nbhd(ii,jj).eq.1)io=2
                end if
                varnew(il,io) = varnew(il,io)
     +          -(v1-v0)*(v1-v0) + (v2-v0)*(v2-v0)
              end if
            end if

c-----------------------------------------------------------------------
c     Update the variogram near the second point (positive lag):
c-----------------------------------------------------------------------
            ii = i2 + ixl(il)
            jj = j2 + iyl(il)
            kk = k2 + izl(il)
            if(ii.ge.1.and.ii.le.nx.and.
     +         jj.ge.1.and.jj.le.ny.and.
     +         kk.ge.1.and.kk.le.nz) then
              if(ii.ne.i1.or.jj.ne.j1.or.kk.ne.k1) then
                v0= var(ii,jj,kk)
                io=1
                if (twopar) then
                 if(nbhd(i1,j1).eq.1.and.cond(ii,jj,kk))io=2
                 if(cond(i1,j1,k1).and.nbhd(ii,jj).eq.1)io=2
                 if(nbhd(i1,j1).eq.1.and.nbhd(ii,jj).eq.1)io=2
                end if
                varnew(il,io) = varnew(il,io)
     +          -(v2-v0)*(v2-v0) + (v1-v0)*(v1-v0)
              end if
            end if

c-----------------------------------------------------------------------
c     Update the variogram near the second point (negative lag):
c-----------------------------------------------------------------------
            ii = i2 - ixl(il)
            jj = j2 - iyl(il)
            kk = k2 - izl(il)
            if(ii.ge.1.and.ii.le.nx.and.
     +         jj.ge.1.and.jj.le.ny.and.
```

```
     +       kk.ge.1.and.kk.le.nz) then
             if(ii.ne.i1.or.jj.ne.j1.or.kk.ne.k1) then
               v0 = var(ii,jj,kk)
               io=1
               if (twopar) then
                 if(nbhd(i1,j1).eq.1.and.cond(ii,jj,kk))io=2
                 if(cond(i1,j1,k1).and.nbhd(ii,jj).eq.1)io=2
                 if(nbhd(i1,j1).eq.1.and.nbhd(ii,jj).eq.1)io=2
               end if
               varnew(il,io) = varnew(il,io)
     +         -(v2-v0)*(v2-v0) + (v1-v0)*(v1-v0)
             end if
           end if

 20      continue
c
c Compute the objective function and return:
c
        objnew = 0.0d0
        do 30 il=1,nlag
        do 30 io=1,part
           act   = varnew(il,io)/divnew(il,io)
           objnew = objnew + (varmod(il)-act)
     +                   * (varmod(il)-act)
     +                   * sclfac(il)
           gammanew(il,io)=act
           if (noisop.eq.1) then
             if (iyl(il).eq.0) gammah(il,io)=act
             if (ixl(il).eq.0) gammav(il,io)=act
           end if
 30      continue

        if(nswap.le.50.and.lagfl(1:10).ne.'nodata.dat')
       +  write(llag,*) ' Obj = ',objnew
        if (nswap.eq.intervar) then
          open(999,file='varmid.dat',status='unknown')
          do il=1,nlag
            dx = dble(ixl(il)) * xsiz
            dy = dble(iyl(il)) * ysiz
            dz = dble(izl(il)) * zsiz
            dx = sqrt(dx*dx+dy*dy+dz*dz)
            write(999,199)il,dx,varmod(il)
          end do
          close(999)
        end if
 199    format(i2,f10.4,2x,3f12.4)
 9901   format(2x,i2,1x,2(2(f4.1,2x),2(f7.4,2x),2(f4.1,2x),f7.4),
       +     3(f12.4,2x))
        return
        end

c-----------------------------------------------------------------------------------------------------------------------
        real*8 function cova3(x1,y1,z1,x2,y2,z2,nst,c0,it,cmax,cc,aa,
       +                ang1,ang2,ang3,anis1,anis2,first)
c
c            Covariance Between Two Points (3-D Version)
c          *****************************************
c
c This function returns the covariance associated with a variogram model that is specified by a nugget effect and possibly four
c different nested varigoram structures.  The anisotropy definition can be different for each of the nested structures (spherical,
c exponential, gaussian, or power).
c
c INPUT VARIABLES:
c
c  x1,y1,z1    Coordinates of first point
c  x2,y2,z2    Coordinates of second point
c  nst         Number of nested structures (max. 4).
c  c0          Nugget constant (isotropic).
```

```
c  cmax        Maximum variogram value needed for kriging when using power model.  A unique value of cmax is used for all
c               nested structures which use the power model.  therefore, cmax should be chosen large enough to account for the
c              argest single structure which uses the power model.
c cc(nst)      Multiplicative factor of each nested structure.  slope for linear model.
c  aa(nst)     Parameter "a" of each nested structure.
c  it(nst)     Type of each nested structure:
c               1. spherical model of range a;
c               2. exponential model of parameter a; i.e. practical range is 3a
c               3. gaussian model of parameter a; i.e. practical range is a*sqrt(3)
c               4. power model of power a (a must be gt. 0  and lt. 2).  if linear model, a=1,c=slope.
c  ang1(nst)      Azimuth angle for the principal direction of continuity (measured clockwise in degrees from Y)
c  ang2(nst)      Dip angle for the principal direction of continuity (measured in negative degrees down)
c  ang3(nst)      Third rotation angle to rotate the two minor directions around the principal direction.  A positive angle acts
c                 clockwise while looking in the principal direction.
c  anis1(nst)     Anisotropy (radius in minor direction at 90 degrees from ang1 divided by the principal radius  in direction ang1)
c  anis2(nst)     Anisotropy (radius in minor direction at 90 degrees vertical from "ang1" divided by the principal radius in direction
c                 "ang1")
c  first          A logical variable which is set to true if the direction specifications have changed – causes  the rotation matrices to be
c                  recomputed.

c
c OUTPUT VARIABLES: returns "cova3" the covariance obtained from the variogram model.
c
c
c NO EXTERNAL REFERENCES:
c-----------------------------------------------------------------------------------------------------------------------------------------
    implicit real*8(a-h,o-z)
    parameter(DTOR=3.14159265d0/180.d0,EPSLON=1.0d-20)
    real*8   aa(*),cc(*),ang1(*),ang2(*),ang3(*),anis1(*),anis2(*),
   +        maxcov
    integer  it(*)
    logical  first
    save     maxcov
c
c The first time around, re-initialize the cosine matrix for the variogram structures:
c
    if(first) then
        maxcov = c0
        do 1 is=1,nst
            if(it(is).eq.4) then
                maxcov = maxcov + cmax
            else
                maxcov = maxcov + cc(is)
            endif
 1      continue
    endif
c
c Check for very small distance:
c
    hsqd = sqdist(x1,y1,z1,x2,y2,z2,ang1(1),ang2(1),ang3(1),
   +          anis1(1),anis2(1))
    if(hsqd.lt.EPSLON) then
        cova3 = maxcov
        return
    endif
c
c Non-zero distance, loop over all the structures:
c
    cova3 = 0.0d0
    do 2 is=1,nst
c
c Compute the appropriate structural distance:
c
        if(is.ne.1) hsqd = sqdist(x1,y1,z1,x2,y2,z2,ang1(is),
   +                 ang2(is),ang3(is),anis1(is),anis2(is))
        h = sqrt(hsqd)
        if(it(is).eq.1) then
c
```

```
c Spherical model:
c
            hr = h/aa(is)
            if(hr.ge.1.0d0) go to 2
            cova3 = cova3 + cc(is)*(1.-hr*(1.5d0-.5d0*hr*hr))
          else if(it(is).eq.2) then
c
c Exponential model:
c
            cova3 = cova3 +cc(is)*dexp(-h/aa(is))
          else if(it(is).eq. 3) then
c
c Gaussian model:
c
            hh=-(h*h)/(aa(is)*aa(is))
            cova3 = cova3 +cc(is)*dexp(hh)
          else
c
c Power model:
c
            cov1  = cmax - cc(is)*(h**aa(is))
            cova3 = cova3 + cov1
          endif
 2    continue
      return
      end
c
c-----------------------------------------------------------------------------------------------------------------------------
c
      real*8 function sqdist(x1,y1,z1,x2,y2,z2,ang1,ang2,ang3,anis1,
     + anis2)
c                  Anisotropic Distance Calculation
c              *********************************
c
c This routine calculates the anisotropic distance between two points given the coordinates of each point and a definition of the
c anisotropy. The components of the vector in the rotated coordinates are calculated and then the squared anisotropic distance is
c calculated.
c
c
c INPUT VARIABLES:
c
c  x1,y1,z1        Coordinates of first point
c  x2,y2,z2        Coordinates of second point
c  ang1            Azimuth angle for the principal direction of continuity (measured clockwise in degrees from Y)
c  ang2            Dip angle for the principal direction of continuity (measured in negative degrees down)
c  ang3            Third rotation angle to rotate the two minor directions around the principal direction.  A positive angle acts clockwise
c                  while looking in the principal direction.
c  anis1           Anisotropy (radius in minor direction at 90 degrees from ang1 divided by the principal radius in direction ang1)
c  anis2           Anisotropy (radius in minor direction at 90 degrees vertical from "ang1" divided by the principal radius in direction
c                  "ang1")
c
c OUTPUT VARIABLES:
c
c  sqdist          The squared distance accounting for the anisotropy and the rotation of coordinates (if any).
c
c PROGRAM NOTES:
c
c    1. The program converts the input (ang1,dip,plg) to three angles which make more mathematical sense:
c
c       alpha   angle between the major axis of anisotropy and the E-W axis. Note: Counter clockwise is positive.
c       beta    angle between major axis and the horizontal plane. (The dip of the ellipsoid measured positive down)
c       theta   angle of rotation of minor axis about the major axis  of the ellipsoid.
C
c NO EXTERNAL REFERENCES
c
c
c Author: C. Deutsch                    Date: July 1989
c-----------------------------------------------------------------------------------------------------------------------------
```

```
      implicit real*8(a-h,o-z)
      parameter(DEG2RAD=3.14159265d0/180.d0)
      real*8   rmatrx(3,3)
      save     rmatrx,ang1o,ang2o,ang3o,anis1o,anis2o
c
c Compute rotation matrix only if required:
c
      if(ang1.ne.ang1o.or.ang2.ne.ang2o.or.ang3.ne.ang3o.or.
     +  anis1.ne.anis1o.or.anis2.ne.anis2o) then
          ang1o  = ang1
          ang2o  = ang2
          ang3o  = ang3
          anis1o = anis1
          anis2o = anis2
          if(ang1.ge.0.d0.and.ang1.lt.270.d0) then
              alpha = (90.0d0  - ang1) * DEG2RAD
          else
              alpha = (450.0d0  - ang1) * DEG2RAD
          endif
          beta  = -1.0d0 * ang2 * DEG2RAD
          theta =      ang3 * DEG2RAD
          cosa  = cos(alpha)
          cosb  = cos(beta)
          cost  = cos(theta)
          sina  = sin(alpha)
          sinb  = sin(beta)
          sint  = sin(theta)
          rmatrx(1,1) =          (cosb * cosa)
          rmatrx(1,2) =          (cosb * sina)
          rmatrx(1,3) =          (-sinb)
          rmatrx(2,1) = (1.0d0/anis1)*(-cost*sina + sint*sinb*cosa)
          rmatrx(2,2) = (1.0d0/anis1)*(cost*cosa + sint*sinb*sina)
          rmatrx(2,3) = (1.0d0/anis1)*( sint * cosb)
          rmatrx(3,1) = (1.0d0/anis2)*(sint*sina + cost*sinb*cosa)
          rmatrx(3,2) = (1.0d0/anis2)*(-sint*cosa + cost*sinb*sina)
          rmatrx(3,3) = (1.0d0/anis2)*(cost * cosb)
      endif
c
c Compute component distance vectors and the squared distance:
c
      dx = x1 - x2
      dy = y1 - y2
      dz = z1 - z2
      sqdist = 0.0d0
      do 1 i=1,3
          temp   = rmatrx(i,1)*dx + rmatrx(i,2)*dy + rmatrx(i,3)*dz
          sqdist = sqdist + temp*temp
 1    continue
      return
      end

c-----------------------------------------------------------------------
c               Order of magnitude of a number
c  Argument
c       x - input number
c     order - order of magnitude of x
c-----------------------------------------------------------------------

      integer function order(x)
      real*8 x,div

      n=0
      if(x.eq.0.d0) then
        order=0
        return
      end if
 67   if(x.gt.1.d0) div=x/(10.d0**n)
      if(x.lt.1.d0) div=x*10.d0**n
```

```
    if(div.lt.10.d0.and.div.ge.1.d0) then
      goto 68
    else
      n=n+1
      goto 67
    end if
68  if(x.gt.1.d0)order=n
    if(x.lt.1.d0)order=-n
    return
    end
```

C SUBROUTINE 'SASIMM' (the SA algorithm)

```
program sasimm
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C                                                                                        %
C Copyright (C) 1992 Stanford Center for Reservoir Forecasting.  All  rights reserved.    %
CDistributed with: C.V. Deutsch and A.G. Journel.                                        %
C ``GSLIB: Geostatistical Software Library and User's Guide," Oxford  University Press, New York, 1992.   %
C                                                                                        %
C The programs in GSLIB are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY.  No author or   %
C distributor accepts  responsibility to anyone for the consequences of using them or for whether they serve any particular purpose   %
C or work at all, unless he says so in writing.  Everyone is granted permission to copy, modify and redistribute the programs in   %
C GSLIB, but only under the condition that this notice and the above copyright notice remain intact.   %
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c-----------------------------------------------------------------------------------------------------------
c
c          Conditional Simulation of a 3-D Rectangular Grid
c       *************************************************
c
c This is a template driver program for GSLIB's "sasim" subroutine. 3-D realizations with a given autocovariance model and
C conditional to input data are created.  The conditional simulation is achieved by modifying an initially uncorrelated image.
c
c The program is executed with no command line arguments.  The user will be prompted for the name of a parameter file.  The
C parameter file is described in the documentation (see the example sasim.par) and should contain the following information:
c
c  -  Name of a data file of conditioning data (GEOEAS format)
c  -  column numbers for x, y, z, and variable
c  -  Minimum acceptable value (used to flag missing values)
c  -  If a standard Normal deviate is to be simulated set to 1
c  -  Name of a data file for non-parametric distribution
c  -  column numbers for variable and weight
c  -  option and parameter for the lower tail
c  -  option and parameter for the upper tail
c  -  An output file (may be overwritten)
c  -  A output file for variograms (may be overwritten)
c  -  The debugging level (integer code - larger means more)
c  -  A file for the debugging output
c  -  Whether or not to use an automatic annealing schedule (0=auto)
c  -  annealing schedule
c  -  Whether a one part or a two part objective function is used
c  -  Random Number Seed
c  -  The number of simulations
c  -  X grid definition (number, minimum, size): nx,xmn,xsiz
c  -  Y grid definition (number, minimum, size): ny,ymn,ysiz
c  -  Z grid definition (number, minimum, size): nz,zmn,zsiz
c  -  The maximum number of lags to be considered
c  -  Search Anisotropy
c  -  Variogram Definition:  number of structures(nst), nugget, and whether or not to renormalize sill to the variance(0=auto)
c  -  the next "nst*2" lines require:
c     First line:
c       a) an integer code for variogram (1=sph,2=exp,3=gaus,4=pow)
c       b) "a" parameter (range except for power model)
c       b) "c" parameter (contribution except for power model).
c     Second line:
c       a) azimuth principal direction (measured clockwise from Y).
c       b) dip of principal direction (measured negative down from X).
```

```
c      c) a third rotation of the two minor directions about the principal direction. This angle acts counterclocwise
c         when looking in the principal direction.
c      Two anisotropy factors are required to complete the definition of the geometric anisotropy of each nested structure:
c      d) radius in minor direction at 90 degrees from the principal direction divided by the principal radius.
c      e) radius in minor direction at 90 degrees vertical from the  principal direction divided by the principal radius.
c
c The output file will be a GEOEAS file containing the simulated values The file is ordered by x,y,z, and then simulation (i.e., x cycles
c fastest, then y, then z, then simulation number).
c
c Original: C.V. Deutsch                    Date: August 1990
c
c 1/20/97 : Updated by Tai-Sheng Liou in the subroutines
c         READPARM - Read mean and variance of ln(k) and ITRANS
c         INITMOD - Change the calling arguments of GINV
C         GINV - Transform N(0,1) to LN(m,s) depending the flag ITRANS. The purpose of this change is to generate a parametric
c         realization of log-normally distributed variate which is then combined with the conditioning data in file 'condfl'
c-----------------------------------------------------------------------------------------------------------------------------------------
      include  'metro.inc'
      character*1 chr1L,chr1R,chr2L,chr2R

      open(500,file='adasp.dat',status='unknown')
      open(550,file='imgasp.dat',status='unknown')
      open(600,file='aspnbhd.dat',status='unknown')
C
c     call timer(itime0)
C
c Read the Parameter File and the Data:
       call readparm
c
C Establish the number of lags to keep
      call getlag
C
c Loop over all the simulations:
C
      do 1 isim=1,nsim
c
c Initialize an image and the statistics :
c
          call initmod
          write(*,*)
          write(ldbg,*)
          write(*,20) ymean,ystd,pcut,paspsave,ptarget,
     +              cutsave,xcut,aspcut,xcut0
          write(ldbg,20) ymean,ystd,pcut,paspsave,ptarget,
     +              cutsave,xcut,aspcut,xcut0
20         format(/' Ensemble staistics :'/
     +         ' Mean of lnk              = ',f6.2/
     +         ' S.t.d. of lnk            = ',f6.2/
     +         ' Initial cutoff probability = ',f6.4/
     +         ' Iterated cutoff probability= ',f6.4/
     +         ' Target cutoff probability  = ',f5.3/
     +         ' Initial cutoff PM value    = ',f7.4/
     +         ' Iterated cutoff PM value   = ',f7.4/
     +         ' Asperity contact         = ',f6.2/
     +         ' Minimum PM value         = ',f6.2//)
          write(ldbg,*)
c
c Call sasim for the simulation:
c
          call sasim
c
c Write the Simulated results, close the output files, and stop:
c
          write(lout,*) 'Permeability field from simulated annealing'
          write(lout,*) 4
          write(lout,999) 'X','X',nx,'X',xmn,'X',xsiz
          write(lout,999) 'Y','Y',ny,'Y',ymn,'Y',ysiz
          write(lout,999) 'Z','Z',nz,'Z',zmn,'Z',zsiz
```

```
999     format(A1,' location N',A1,'=',i4,1x,2(1x,'D',A1,'=',f6.2))
        write(lout,*) 'Permeability field'
        do 2 iz=1,nz
        do 2 iy=1,ny
        do 2 ix=1,nx
           xx=xmn+xsiz*dble(ix-1)
           yy=ymn+ysiz*dble(iy-1)
           zz=zmn+zsiz*dble(iz-1)
           if(cond(ix,iy,iz)) then
             if (var(ix,iy,iz).eq.xcut0) var(ix,iy,iz)=0.0d0
             write(lout,101) xx,yy,zz,var(ix,iy,iz)
           else
             write(lout,102) xx,yy,zz,var(ix,iy,iz)
           endif
c
c 7/16/97 : Write additional asperity contacts and z'<=5.0 to addasp.dat
c
           if(var(ix,iy,iz).eq.0.0d0) then
            if(.not.cond(ix,iy,iz))write(500,102) xx,yy,zz,0.0
           end if
           if (var(ix,iy,iz).gt.0.0d0.and.
     +       var(ix,iy,iz).le.5.0d0)
     +       write(500,102) xx,yy,zz,var(ix,iy,iz)
C
c
c 10/8/97 : Write permeability modifier, excluding the conditioning asperity contact data, in the neighborhood
c
         if (nbhd(ix,iy).gt.0)
     +     write(600,102) xx,yy,zz,var(ix,iy,iz)
 2      continue
 101    format(3(f8.2,2x),f12.4,' c')
 102    format(3(f8.2,2x),f12.4)

c
c Calculate the average permeability for two kinds of neighborhood
c
c First, assign nbhd(i,j)=1 for defnbhd=1 only  and   nbhd(i,j)=2 for defnbhd=1 and defnbhd=2
c
    size=real(neighbor)
    do 12 i=1,nx
    do 12 j=1,ny
    do 12 k=1,nz
      if (cond(i,j,k)) then
        do ii=-neighbor,neighbor
          do jj=-neighbor,neighbor
          ix=i+ii
          iy=j+jj
          iz=k
          dx=real(ii)
          dy=real(jj)
          dist=sqrt(dx*dx+dy*dy)
          if (ix.ge.1.and.ix.le.nx.and.iy.ge.1.and.iy.le.ny.and.
     +       iz.ge.1.and.iz.le.nz) then
            if(.not. cond(ix,iy,iz).and.nbhd(ix,iy).eq.0)
     +       nbhd(ix,iy)=1
            if(nbhd(ix,iy).eq.1.and.dist.le.size) nbhd(ix,iy)=2
          end if
          end do
        end do
      end if
12   continue
c
c Second, calculate avg. PM for two kinds of neighborhood
c
     if (neighbor.ne.0) then
       zsum1=0.0
       zsum2=0.0
       noz1=0
```

```
        noz2=0
        do 3 ix=1,nx
        do 3 iy=1,ny
        do 3 iz=1,nz
          if (nbhd(ix,iy).eq.2) then
            zsum2=zsum2+var(ix,iy,iz)
            noz2=noz2+1
            zsum1=zsum1+var(ix,iy,iz)
            noz1=noz1+1
          end if
          if (nbhd(ix,iy).eq.1) then
            zsum1=zsum1+var(ix,iy,iz)
            noz1=noz1+1
          end if
3       continue
        if (defnbhd.eq.1) then
          chr2L='('
          chr2R=')'
        else
          chr1L='('
          chr1R=')'
        end if
        write(ldbg,*)
        write(ldbg,*) ' NBHD = ',defnbhd
        write(ldbg,50)chr1L,noz1,zsum1,zsum1/dble(noz1),chr1R,
     +              chr2L,noz2,zsum2,zsum2/dble(noz2),chr2R
50      format(/,
     +2x,A1,'First nbhd : Nb1= ',i5,' Sum1= ',f9.4,' Avg= ',f8.5,A1/
     +2x,A1,'Second nbhd: Nb2= ',i5,' Sum2= ',f9.4,' Avg= ',f8.5,A1)
        end if
c
c End loop over all simulations:
c
 1    continue
c     call timer(itime1)
      itime=itime1-itime0
      time=real(itime)/100.0
      ihr=int(time/3600.0)
      imin=int((time-real(ihr)*3600.0)/60.0)
      sec=time-real(ihr)*3600.0-real(imin)*60.0
      write(ldbg,5) ihr,imin,sec
      write(*,5) ihr,imin,sec
5     format(/1x,'Elapsed time = ',i2,' hours ,',i3,' mins, ',f5.2,
     +     ' secs')

      close(lout)
      close(lvar)
      close(ldbg)
      write(*,*) 'Finished SASIM: simulated results in ',outfl
      write(*,*) '              variogram output in  ',varfl
      write(*,*) '              debugging output in  ',dbgfl
      write(*,*)
      stop
      end



      subroutine readparm
c-----------------------------------------------------------------------
c
c                Initialization and Read Parameters
c               **********************************
c
c The input parameters are read from a file name provided from standard input (a default name will be tried if none is keyed in by the
C user).
c
c The complete 3-D field is then filled in with values drawn at random from either a standard normal distribution or some distribution
c specified in a non-parametric way (i.e., a series of values and associated weights) with possibly a parametric option to treat values
```

```
c in the tails of the distribution.
c
c Conditioning data is then read in (if available) and assigned to the nearest node if within the grid network.
c
c Error checking is performed and the statistics of both the initialrealization and conditioning data are written to the debugging file.
c
c
c
c Original: C.V. Deutsch                    Date: July 1990
c-----------------------------------------------------------------------------------------------------------------------------------------
      include  'metro.inc'
      parameter(MV=20)
      real*8     val(MV)
      logical   testfl,image
      character str*40,title*80
c
c Unit numbers:
c
      lin  = 1
      lout = 2
      ldbg = 3
      lvar = 4
      lhvar = 5
      lvvar = 6
      llag  = 7
      limagein= 8
      limageout=9
c
c Open the input file 'sasim.par'
c
      open(lin,file='sasim.par',status='OLD')
c
c Find Start of Parameters:
c
 1    read(lin,'(a4)',end=97) str(1:4)
      if(str(1:4).ne.'STAR') go to 1
c
c Read Input Parameters:
c

      read(lin,'(a40)',err=97) imageinfl
      if (imageinfl(1:10).ne.'nodata.dat')
     +   write(*,*) 'Initial image file   : ',imageinfl
      read(lin,'(a40)',err=97) imageoutfl
      write(*,*) 'Output image file    : ',imageoutfl
      read(lin,'(a40)',err=97) condfl
      write(*,*) 'Conditioning data file: ',condfl
      read(lin,*,err=97)       ixloc,iyloc,izloc,ivrl
      read(lin,*,err=97)       tmin,tmax
      read(lin,*,err=97)       igauss
      read(lin,'(a40)',err=97) datafl
      read(lin,*,err=97)       ivr,iwt
      read(lin,*,err=97)       zmin,zmax
      read(lin,*,err=97)       ltail,ltpar
      read(lin,*,err=97)       utail,utpar
      read(lin,'(a40)',err=97) outfl
      write(*,*) 'Output file:          ',outfl
      read(lin,'(a40)',err=97) varfl
      write(*,*) 'Overall variogram output file:  ',varfl
      read(lin,'(a40)',err=97) horvarfl
      write(*,*) 'Horizontal variogram output file:  ',horvarfl
      read(lin,'(a40)',err=97) vervarfl
      write(*,*) 'Vertical variogram output file:   ',vervarfl
      read(lin,*,err=97)       idbg,report,intervar
      read(lin,'(a40)',err=97) dbgfl
      write(*,*) 'Debug file:           ',dbgfl
      open(ldbg,file=dbgfl,status='UNKNOWN')
      write(ldbg,*) 'Conditioning data file: ',condfl
```

```
      read(lin,'(a40)',err=97) lagfl
      write(*,*) 'Lag file:            ',lagfl
      write(*,*)'************************************************'
      read(lin,*,err=97) isas
      write(*,111) isas
      write(ldbg,111) isas
  111 format(/' Annealing schedule : ',i2/,
     + ' ( 0:user, 1:default, 2:fast, 3:very fast ) ')
      read(lin,*,err=97) (sas(i),i=1,6)
      read(lin,*,err=97)      part
      read(lin,*,err=97)      seed
      read(lin,*,err=97)      nsim
      read(lin,*,err=97)      nx,xmn,xsiz
      read(lin,*,err=97)      ny,ymn,ysiz
      read(lin,*,err=97)      nz,zmn,zsiz
      read(lin,*,err=97)      nlag,neighbor
c
c 1/23/97 : Select the annealing schedule
c     0 - User supplied
c     1 - Default
c     2 - Fast
c     3 - Very fast
c Tai-Sheng Liou (Deutsch and Cockerham, 1994)
c
      sas(3)=sas(3)*dble(nx*ny*nz)
      sas(4)=sas(4)*dble(nx*ny*nz)
      if(isas.eq.1) then
          sas(1) = 1.0d0
          sas(2) = 0.1d0
          sas(3) = 100.d0*dble(nx*ny*nz)
          sas(4) = 10.d0*dble(nx*ny*nz)
          sas(5) =  3.d0
          sas(6) = 0.001d0
      elseif(isas.eq.2) then
          sas(1) = 1.0d0
          sas(2) = 0.05d0
          sas(3) = 50.d0*dble(nx*ny*nz)
          sas(4) = 5.d0*dble(nx*ny*nz)
          sas(5) =  3.d0
          sas(6) = 0.001d0
       elseif(isas.eq.3) then
          sas(1) = 0.5d0
          sas(2) = 0.01d0
          sas(3) = 10.d0*dble(nx*ny*nz)
          sas(4) = 2.d0*dble(nx*ny*nz)
          sas(5) =  3.d0
          sas(6) = 0.001d0
      endif
      write(*,112) (sas(i),i=1,6),part
      write(ldbg,112) (sas(i),i=1,6),part
  112 format(' User set schedule : '/
     +      ' T0      = ',f5.1,/
     +      ' T factor = ',f5.1,/
     +      ' Kmax    = ',e7.1,/
     +      ' Kaccept = ',e7.1,/
     +      ' S       = ',f5.1,/
     +      ' Omin    = ',e7.1,/
     +      ' Part    = ',i2)
      write(*,*)'************************************************'
      read(lin,*,err=97)      nst,c0,isill
      sill = c0
      write(*,100) isill,nst,c0
      if(nst.le.0) then
          write(*,9997) nst
 9997     format(' nst must be at least 1, it has been set to ',i4,/,
     +          ' The c or a values can be set to zero')
          stop
      endif
```

```
c
c 1/25/97 : noisop=0 for isotropic variogram;  noisop=1 for anisotropic variograms
c
      noisop=0
      do 3 i=1,nst
        read(lin,*,err=97) it(i),aa(i),cc(i)
        sill = sill + cc(i)
        read(lin,*,err=97) ang1(i),ang2(i),ang3(i),anis1(i),anis2(i)
        if (anis1(i).ne.1.or.anis2(i).ne.1) noisop=1
        write(ldbg,*)
        write(ldbg,*) 'Semi-variogram model :'
        If (it(i).eq.1) write(ldbg,*) 'Spherical model'
        if (it(i).eq.2) write(ldbg,*) 'Exponential model'
        if (it(i).eq.3) write(ldbg,*) 'Gaussian model'
        if (it(i).eq.4) write(ldbg,*) 'Power model'
        if (noisop.eq.0) write(ldbg,*) 'isotropic model with'
        if (noisop.eq.1) write(ldbg,*) 'anisoptropic model with'
        write(ldbg,20) sill,aa(i)
        if (noisop.eq.1) then
          write(ldbg,21) ang1(i),ang2(i),ang3(i),anis1(i),anis2(i)
        end if
 3    continue
20    format('Sill = ',f8.2/'Longitudinal correlaiton length = ',f8.2)
21    format('anisotropic angle1 = ',f8.2/
     +       'anisotropic angle2 = ',f8.2/
     +       'anisotropic angle3 = ',f8.2/
     +       'anisotropic ratio1 = ',f8.2/
     +       'anisotropic ratio2 = ',f8.2)
c
c 1/21/97 : The following lines are added by Tai-Sheng Liou
c iedge=(0)1-(not)correct edge-effect with weighting factor=wfedge
c icond=(0)1-(not)correct discontinuity-effect with weighting factor=wfcond
c itrans=(0)1-(not)transform N(0,1) to LN(xmean,xstd) [y=lnx]
c itrans=2  - transform N(0,1) to LN(xmean,xstd) but do not shift the LOGNORMAL data
c noiter    - maximum number of iteration for calculating the sample statistics in order to have statistics as close to ensemble statistics
c             as possible
c
      read(lin,*) iedge,wfedge
      read(lin,*) icond,wfcond
      write(*,*)'***********************************************'
      write(*,*)
c     write(*,*) 'Weighting factors :'
c     if (iedge.eq.0) write(*,*)'Edge effect not weighted'
c     if (iedge.eq.1) write(*,21) wfedge
c     if (iedge.eq.0) write(ldbg,*)'Edge effect not weighted'
c     if (iedge.eq.1) write(ldbg,21) wfedge
c21   format(' Edge factor        = ',f6.2)
c     if (icond.eq.0) write(*,*) 'Discontinuity effect not weighted'
c     if (icond.eq.1) write(*,22) wfcond
c     if (icond.eq.0) write(ldbg,*) 'Discontinuity effect not weighted'
c     if (icond.eq.1) write(ldbg,22) wfcond
c22   format(' Discontinuity factor = ',f6.2)
      write(*,23) nlag,neighbor
      write(ldbg,23) nlag,neighbor
23    format(' Total # of lags     = ',i4/
     +       ' # of neighborhood   = ',i4)
      write(*,*)'***********************************************'
      read(lin,*) ymean,ystd,itrans
      read(lin,*) pcut,aspcut,xcut,ptarget
c
c Read which annealing algorithm should be used:
c 0 - standard Metropolis algorithm, 1- modified Metropolis algorithm
c
      read(lin,*) metro
      write(ldbg,*)
      if (metro.eq.0) then
        write(*,*) 'Using standard Metropolis algorithm'
        write(ldbg,*) 'Using standard Metropolis algorithm'
```

```
      elseif(metro.eq.1) then
        write(*,*) 'Using modified Metropolis algorithm'
        write(ldbg,*) 'Using modified Metropolis algorithm'
      else
        write(*,*) 'Type of SA algorithm is not specified'
        stop
      end if
      read(lin,*) defnbhd
      write(ldbg,*)
      if (defnbhd.eq.1) then
        write(*,*) 'The neighborhood is defined as a square'
        write(ldbg,*) 'The neighborhood is defined as a square'
      elseif (defnbhd.eq.2) then
        write(*,*) 'The neighborhood is a square plus four ears'
        write(ldbg,*) 'The neighborhood is a square plus four ears'
      else
        if(metro.eq.1) then
          write(*,*) 'Neighborhood is not defined'
          stop
        end if
      end if
      write(ldbg,*)
      read(lin,*) imod
      if(imod.lt.1.or.imod.gt.4) then
        write(*,*) 'Choose the type of Modified Metropolis',
     +           ' algorithm, i.e., imod=1 or 4'
        stop
      end if
      write(*,*) 'Using MOD',imod
      write(ldbg,*) 'Using MOD',imod
      read(lin,*) ifield
      if(ifield.lt.0.or.ifield.gt.1) then
        write(*,*)'Enter 0 (No) or 1 (Yes) to print out evolving files'
      end if
      if(ifield.eq.0) then
        write(*,*) 'Do not generate internal files of PM fields'
        write(ldbg,*) 'Do not generate internal files of PM fields'
      else
        write(*,*) 'Generate internal files of PM fields'
        write(ldbg,*) 'Generate internal files of PM fields'
      end if
c
      close(lin)
  100 format(/,' Reset sill: ',i2,/,
     +       '       number of structures = ',i3,/,
     +       '       nugget effect        = ',f8.4)
  101 format( '       type of structure ',i3,' = ',i3,/,
     +       '       aa parameter         = ',f12.4,/,
     +       '       cc parameter         = ',f12.4)
  102 format( '       ang1, ang2, ang3     = ',3f6.2,/,
     +       '       anis1, anis2         = ',2f12.4)
c
c Reset the annealing schedule if automatic timing is being used:
c
      if(part.eq.1) then
          twopar = .false.
      else
          twopar = .true.
      endif
      if(nz.le.1.or.izloc.le.0) then
          only2d = .true.
      else
          only2d = .false.
      endif
c
c Perform some quick error checking:
c
      if(nx.gt.MAXX) stop 'nx is too big - modify .inc file'
```

```
      if(ny.gt.MAXY) stop 'ny is too big - modify .inc file'
      if(nz.gt.MAXZ) stop 'nz is too big - modify .inc file'
      if(nlag.gt.MAXLAG) stop 'nlag is too big - modify .inc file'
      if(nst.gt.MAXNST) stop 'nst is too big - modify .inc file'
c
c Open the debugging and output files:
c
      open(lvar,file=varfl,status='unknown')
      open(lhvar,file=horvarfl,status='UNKNOWN')
      open(lvvar,file=vervarfl,status='unknown')
      open(lout,file=outfl,status='UNKNOWN')

      if(lagfl(1:10).ne.'nodata.dat')
     +  open(llag,file=lagfl,status='unknown')
      if (imageinfl(1:10).ne.'nodata.dat')
     +  open(limagein,file=imageinfl,status='unknown')
      if (imageoutfl(1:10).ne.'nodata.dat')
     +  open(limageout,file=imageoutfl,status='unknown')
c
c If possible read in the cdf ("cut" and "cdf" arrays) to use as the distribution to initialize the realization:
c
      title = 'SASIM SIMULATIONS:                '//
     +        '                                  '
      if(igauss.eq.0) then
          inquire(file=datafl,exist=testfl)
          if(.not.testfl) then
              write(*,*) 'ERROR file ',datafl,' does not exist!'
              write(*,*) '  you need a univariate distribution'
              write(*,*) '  unless you want a Gaussian distribution'
              stop
          endif
          ncdf = 0.0d0
          ccdf = 0.0d0
          open(lin,file=datafl,status='OLD')
          read(lin,'(a60)',err=98) title(21:80)
          read(lin,*,err=98) nvari
          do 4 i=1,nvari
 4         read(lin,*,err=98)
 5         read(lin,*,end=6,err=98) (val(j),j=1,nvari)
          if(val(ivr).lt.tmin.or.val(ivr).ge.tmax) go to 5
          ncdf     = ncdf + 1
          if(ncdf.gt.MAXCUT) then
              write(*,*) ' ERROR: exceeded storage for cdf',ncdf
              stop
          endif
          cut(ncdf) = val(ivr)
          if(iwt.le.0) then
              cdf(ncdf) = 1.0d0
          else
              cdf(ncdf) = val(iwt)
          endif
          ccdf = ccdf + cdf(ncdf)
          go to 5
 6        close(lin)
c
c Turn the (possibly weighted) distribution into a cdf that can be used to initialize all the grid nodes:
c
          call sortem(1,ncdf,cut,1,cdf,c,d,e,f,g,h)
          oldcp = 0.0d0
          cp    = 0.0d0
          ccdf  = 1.0d0 / ccdf
          do 7 i=1,ncdf
              cp     = cp + cdf(i) * ccdf
              cdf(i) =(cp + oldcp) * 0.5d0
              oldcp  = cp
 7        continue
      endif
c
```

```
c Turn all conditioning flags to false:
c
      do 8 ix=1,nx
      do 8 iy=1,ny
      do 8 iz=1,nz
            cond(ix,iy,iz) = .false.
 8    continue
c
c Check to see if a file of conditioning data exists, if it does then read in the data:
c
      inquire(file=condfl,exist=testfl)
      if(testfl) then
            open(lin,file=condfl,status='OLD')
            read(lin,'(a60)',err=99) title(21:80)
            read(lin,*,err=99)       nvari
            nd = 0
            av = 0.0d0
            ss = 0.0d0
            do 9 i=1,nvari
 9           read(lin,'(a40)',err=99) str
c
c Read all the data until the end of the file:
c
 10          read(lin,*,end=11,err=99) (val(j),j=1,nvari)
            if(val(ivrl).le.tmin.or.val(ivrl).gt.tmax) go to 10
            nd  = nd + 1
            av  = av + val(ivrl)
            ss  = ss + val(ivrl)*val(ivrl)
            ix=min0(max0((int((val(ixloc)-xmn)/xsiz+0.5d0)+1),1),nx)
            iy=min0(max0((int((val(iyloc)-ymn)/ysiz+0.5)+1),1),ny)
            iz=min0(max0((int((val(izloc)-zmn)/zsiz+0.5)+1),1),nz)
            if(only2d) iz = 1
            var(ix,iy,iz)  = val(ivrl)
            cond(ix,iy,iz) = .true.
            go to 10
 11         close(lin)
c
c Compute the averages and variances as an error check for the user:
c
            av = av / amax1(dble(nd),1.0)
            ss =(ss / amax1(dble(nd),1.0)) - av * av
            write(ldbg,*) 'Data for SASIM: Variable number ',ivrl
            write(ldbg,*) '  Number of acceptable data  = ',nd
            write(ldbg,*) '  Equal Weighted Average     = ',av
            write(ldbg,*) '  Equal Weighted Variance    = ',ss
      endif
c
c 1/27/97 : Calcuate the number of data points that is not conditioning data (noncond)
c
            noncond=nx*ny*nz-nd
c
c Write a header on the output file and return:
c 1/23/97 : Comment out by Tai-Sheng Liou

c
c     write(lout,105) title
c 105  format(a80,/,'1',/,'simulated value')

      return
c
c Error in an Input File Somewhere:
c

 97   stop 'ERROR in parameter file!'
 98   stop 'ERROR in distribution file!'
 99   stop 'ERROR in data file!'
      end
```

```
      subroutine initmod
c-----------------------------------------------------------------------
c
c                  Initialization of Grid
c                 **********************
c
c-----------------------------------------------------------------------
      include  'metro.inc'
      real*8     randnu(1)
      logical image

c
c Initialize all nodes to some random quantile:
c
      call rand(seed,1,randnu)
      seed = 0

      do 6 ix=1,nx
      do 6 iy=1,ny
      do 6 iz=1,nz
        if(.not.cond(ix,iy,iz)) var(ix,iy,iz)=0.d0
6     continue

c
c 2/4/97 : Read the initial image file or automatically genreate the initial image by either non-parametric distribution or Gaussian
c          distribution,  Tai-Sheng Liou
c
      inquire(file=imageinfl,exist=image)
      if (image) then
        read(limagein,*)
        read(limagein,*) nheader
        do i=1,nheader
          read(limagein,*)
        end do
        do 10 iz=1,nz
        do 10 iy=1,ny
        do 10 ix=1,nx
          read(limagein,*) dummy,dummy,dummy,var(ix,iy,iz)
10      continue
        write(*,15) imageinfl
15      format(' Using ',a15,' as initial image')
        write(*,*)
      else
c
c 1/27/97 :  Draw a Monte Carlo Realization from either a Gaussian distribution (igauss.ne.0)  or a non-parametric distribution
c             (igauss.eq.0),     Tai-Sheng Liou
c
       if (igauss.eq.0) then
        do 1 i=1,nx
        do 1 j=1,ny
        do 1 z=1,nz
c
c Only initialize if not a conditioning datum:
c
          if(cond(i,j,k)) go to 1
          call rand(seed,1,randnu)
          call beyond(ncdf,cut,cdf,zmin,zmax,ltail,
     +         ltpar,utail,utpar,var(i,j,k),randnu(1),ierr)
1       continue
       elseif (igauss.eq.1) then
         call gridxyz(seed)
       else
         call etapdf(seed)
       end if
       end if
c
```

```
c Renormalize the variogram parameters to the variance of the realization if requested:
c
      if(isill.eq.1) then
c
c Get current sill of variogram:
c
            sill = c0
            do 2 i=1,nst
 2          sill = sill + cc(i)
c
c Get variance of realization:
c
            av = 0.0d0
            ss = 0.0d0
            do 3 i=1,nx
            do 3 j=1,ny
            do 3 k=1,nz
                av = av + var(i,j,k)
                ss = ss + var(i,j,k)*var(i,j,k)
 3          continue
            av = av / dble(nx*ny*nz)
            ss = ss / dble(nx*ny*nz) - av*av
c
c Now, scale the variogram parameters:
c
            fac = ss/sill
            c0  = c0 * fac
            do 4 i=1,nst
 4          cc(i)  = cc(i) * fac
c
c Also, scale the varmod array:
c
            do 5 i=1,nlag
 5          varmod(i) = varmod(i) * fac
      endif
c
c Finished getting initial image:
c
      return
      end



      subroutine rand(seed,n,vector)
c-----------------------------------------------------------------------
c
c This random number generator generates random numbers in ]0,1[ Note that if the seed value is zero on the first call, a default value
c of 1369 will be used  in a linear congruential generator to generate 55 odd integers for the array 'itab()'. These values are preserved
c by a common statement, so that they may be used in subsequent calls by setting the seed to zero.If the value of 'seed' is greater than
c zero in a call to the subroutine, then the array 'itab' will be initialized and a new seed value will be returned by the subroutine. Best
c results are obtained by making the initial call with a seed of your choice and then setting the seed to '0' for all subsequent calls.
c
c-----------------------------------------------------------------------
      implicit real*8(a-h,o-z)
      real*8 vector(*)
      common /unusual/itab(55),n1,n2,nseed
      integer rm1,seed
c
c Test to see if 55 odd integers must be generated.
c
      if((seed.gt.0).or.(nseed.lt.1)) then
          nseed = seed
          if(seed.le.0) nseed  = 7931
          do 10 i=1,55
              rm1=mod(nseed*9069,32768)
              if(mod(rm1,2).eq.0) rm1 = rm1-1
              itab(i) = rm1
              nseed = rm1
```

```
10      continue
        n1 = 0
        n2 = 24
      endif
c
c generate "n" random components for the vector "VECTOR"
c
      do 30 i=1,n
         itab(55-n1) = mod(itab(55-n2)*itab(55-n1),32768)
         vector(i) = abs(float(itab(55-n1))/float(32768))
         n1 = mod(n1+1,55)
         n2 = mod(n2+1,55)
30    continue
      if(seed.gt.0) seed=nseed
      return
      end

      subroutine locate(xx,n,is,ie,x,j)
c-------------------------------------------------------------------------------------------------------------------------------
c
c Given an array "xx" of length "n", and given a value "x", this routine returns a value "j" such that "x" is between xx(j) and xx(j+1).
C xx must be monotonic, either increasing or decreasing. j=0 or j=n is returned to indicate that x is out of range.
c
c Modified to set the start and end points by "is" and "ie"
c
c Bisection Concept From "Numerical Recipes", Press et. al. 1986  pp 90.
c-------------------------------------------------------------------------------------------------------------------------------
      implicit real*8(a-h,o-z)
      real*8 xx(n)
c
c Initialize lower and upper methods:
c
      jl = is
      ju = ie
c
c If we are not done then compute a midpoint:
c
10    if(ju-jl.gt.1) then
         jm = (ju+jl)/2
c
c Replace the lower or upper limit with the midpoint:
c
         if((xx(n).gt.xx(1)).eqv.(x.gt.xx(jm))) then
            jl = jm
         else
            ju = jm
         endif
         go to 10
      endif
c
c Return with the array index:
c
      j = jl
      return
      end

      subroutine sortem(ib,ie,a,iperm,b,c,d,e,f,g,h)
c-------------------------------------------------------------------------------------------------------------------------------
c
c                  Quickersort Subroutine
c                 ************************
c
c This is a subroutine for sorting a real array in ascending order. This is a Fortran translation of algorithm 271, quickersort, by R.S.
c Scowen in collected algorithms of the ACM. The method used is that of continually splitting the array into parts such that all
c elements of one part are less than all elements of the other, with a third part in the middle consisting of one element. An element
c with value t is chosen arbitrarily (here we choose the middle element). i and j give the lower and upper limits of the segment being
c split. After the split a value q will have been found such that  a(q)=t and a(l)<=t<=a(m) for all i<=l<q<m<=j. The program then
c performs operations on the two segments (i,q-1) and (q+1,j) as follows The smaller segment is split and the position of the larger
```

```
c segment is stored in the lt and ut arrays.  If the segment to be split contains two or fewer elements, it is sorted and another segment is
c obtained from the lt and ut arrays.  When no more segments remain, the array is completely sorted.
c
c
c INPUT PARAMETERS:
c
c  ib,ie      start and end index of the array to be sorteda
c  a          array, a portion of which has to be sorted.
c  iperm       0 no other array is permuted.
c             1 array b is permuted according to array a
c             2 arrays b,c are permuted.
c             3 arrays b,c,d are permuted.
c             4 arrays b,c,d,e are permuted.
c             5 arrays b,c,d,e,f are permuted.
c             6 arrays b,c,d,e,f,g are permuted.
c             7 arrays b,c,d,e,f,g,h are permuted.
c            >7 no other array is permuted.
c
c  b,c,d,e,f,g,h  arrays to be permuted according to array a.
c
c OUTPUT PARAMETERS:
c
c   a     = the array, a portion of which has been sorted.
c
c   b,c,d,e,f,g,h  =arrays permuted according to array a (see iperm)
c
c NO EXTERNAL ROUTINES REQUIRED:
c
c-----------------------------------------------------------------------------------------------------------------------------------------
implicit real*8(a-h,o-z)
    real*8 a(*),b(*),c(*),d(*),e(*),f(*),g(*),h(*)
c
c The dimensions for lt and ut have to be at least log (base 2) n
c
    integer   lt(64),ut(64),i,j,k,m,p,q
c
c Initialize:
c
    j   = ie
    m   = 1
    i   = ib
    iring = iperm+1
    if (iperm.gt.7) iring=1
c
c If this segment has more than two elements  we split it
c
 10  if (j-i-1) 100,90,15
c
c p is the position of an arbitrary element in the segment we choose the middle element. Under certain circumstances it may be
c advantageous to choose p at random.
c
 15  p   = (j+i)/2
    ta  = a(p)
    a(p) = a(i)
    go to (21,19,18,17,16,161,162,163),iring
 163    th  = h(p)
      h(p) = h(i)
 162    tg  = g(p)
      g(p) = g(i)
 161    tf  = f(p)
      f(p) = f(i)
 16     te  = e(p)
      e(p) = e(i)
 17     td  = d(p)
      d(p) = d(i)
 18     tc  = c(p)
      c(p) = c(i)
 19     tb  = b(p)
```

```
      b(p) = b(i)
 21   continue
c
c Start at the beginning of the segment, search for k such that a(k)>t
c
     q = j
     k = i
 20   k = k+1
     if(k.gt.q)     go to 60
     if(a(k).le.ta) go to 20
c
c Such an element has now been found now search for a q such that a(q)<t starting at the end of the segment.
c
 30   continue
     if(a(q).lt.ta) go to 40
     q = q-1
     if(q.gt.k)     go to 30
     go to 50
c
c a(q) has now been found. we interchange a(q) and a(k)
c
 40   xa   = a(k)
     a(k) = a(q)
     a(q) = xa
     go to (45,44,43,42,41,411,412,413),iring
 413    xh   = h(k)
       h(k) = h(q)
       h(q) = xh
 412    xg   = g(k)
       g(k) = g(q)
       g(q) = xg
 411    xf   = f(k)
       f(k) = f(q)
       f(q) = xf
 41     xe   = e(k)
       e(k) = e(q)
       e(q) = xe
 42     xd   = d(k)
       d(k) = d(q)
       d(q) = xd
 43     xc   = c(k)
       c(k) = c(q)
       c(q) = xc
 44     xb   = b(k)
       b(k) = b(q)
       b(q) = xb
 45   continue
c
c Update q and search for another pair to interchange:
c
     q = q-1
     go to 20
 50   q = k-1
 60   continue
c
c The upwards search has now met the downwards search:
c
     a(i)=a(q)
     a(q)=ta
     go to (65,64,63,62,61,611,612,613),iring
 613    h(i) = h(q)
       h(q) = th
 612    g(i) = g(q)
       g(q) = tg
 611    f(i) = f(q)
       f(q) = tf
 61     e(i) = e(q)
       e(q) = te
```

```
 62     d(i) = d(q)
        d(q) = td
 63     c(i) = c(q)
        c(q) = tc
 64     b(i) = b(q)
        b(q) = tb
 65   continue
c
c The segment is now divided in three parts: (i,q-1),(q),(q+1,j) store the position of the largest segment in lt and ut
c
      if (2*q.le.i+j) go to 70
      lt(m) = i
      ut(m) = q-1
      i = q+1
      go to 80
 70   lt(m) = q+1
      ut(m) = j
      j = q-1
c
c Update m and split the new smaller segment
c
 80   m = m+1
      go to 10
c
c We arrive here if the segment has  two elements we test to see if the segment is properly ordered if not, we perform an interchange
c
 90   continue
      if (a(i).le.a(j)) go to 100
      xa=a(i)
      a(i)=a(j)
      a(j)=xa
      go to (95,94,93,92,91,911,912,913),iring
 913    xh  = h(i)
        h(i) = h(j)
        h(j) = xh
 912    xg  = g(i)
        g(i) = g(j)
        g(j) = xg
 911    xf  = f(i)
        f(i) = f(j)
        f(j) = xf
  91    xe  = e(i)
        e(i) = e(j)
        e(j) = xe
  92    xd  = d(i)
        d(i) = d(j)
        d(j) = xd
  93    xc  = c(i)
        c(i) = c(j)
        c(j) = xc
  94    xb  = b(i)
        b(i) = b(j)
        b(j) = xb
  95 continue
c
c If lt and ut contain more segments to be sorted repeat process:
c
 100  m = m-1
      if (m.le.0) go to 110
      i = lt(m)
      j = ut(m)
      go to 10
 110  continue
      return
      end
```

```
      subroutine beyond(ncut,cut,cdf,zmin,zmax,ltail,ltpar,utail,utpar,
     +                  zval,cdfval,ierr)
c-----------------------------------------------------------------------
c
c                  Go Beyond a Discrete CDF
c                 ************************
c
c This subroutine is a general purpose subroutine to interpolate within and extrapolate beyond discrete points on a conditional CDF. If
c the Z value "zval" is specified then the corresponding CDF value "cdfval" will be computed, if the CDF value "cdfval" is specified
c the corresponding Z value "zval" will be computed.
c
c
c
c INPUT/OUTPUT VARIABLES:
c
c  ncut          number of cutoffs defining the global CDF
c  cut()         real array of the ncut cutoffs
c  cdf()         real array of the global cdf values
c
c  zmin,zmax     minimum and maximum allowable data values
c  ltail         option to handle values in lower tail
c  ltpar         parameter required for option ltail
c  utail         option to handle values in upper tail
c  utpar         parameter required for option utail
c
c  zval          interesting cutoff (if -1 then it is calculated)
c  cdfval        interesting CDF (if -1 then it is calculated)
c
c  ierr          error flag:  0 - no problem
c                             1 - both zval or cdfval can not be
c                                 defined or undefined
c                             2 - invalid parameters
c
c
c Original: C.V. Deutsch                        October 1991
c-----------------------------------------------------------------------

      implicit real*8(a-h,o-z)
      parameter(EPSLON=1.0d-20,UNEST=-1.0d0)
      dimension cut(ncut),cdf(ncut)
      real*8    utpar,ltpar,lambda
      integer   ltail,utail
c
c Figure out what part of distribution: ipart = 0 - lower tail
c                                       ipart = 1 - middle
c                                       ipart = 2 - upper tail
                  ipart = 1
      if(cdfval.le.cdf(1))   ipart = 0
      if(cdfval.ge.cdf(ncut)) ipart = 2
c
c ARE WE IN THE LOWER TAIL?
c
      if(ipart.eq.0) then
          if(ltail.eq.1) then
c
c Straight Linear Interpolation:
c
              powr = 1.0d0
              zval = powint(0.0d0,cdf(1),zmin,cut(1),cdfval,powr)
          else if(ltail.eq.2) then
c
c Power Model interpolation to lower limit "zmin"?
c
              cpow = 1.0d0 / ltpar
              zval = powint(0.0d0,cdf(1),zmin,cut(1),cdfval,cpow)
          else
c
c Error situation - unacceptable option:
```

```
c
            ierr = 2
            return
          endif
      endif
c
c FINISHED THE LOWER TAIL,  ARE WE IN THE MIDDLE?
c
      if(ipart.eq.1) then
c
c Linear interpolation between the rescaled global cdf?
c
          call locate(cdf,ncut,1,ncut,cdfval,ilow)
          ihigh = ilow + 1
          powr  = 1.0d0
          zval  = powint(cdf(ilow),cdf(ihigh),cut(ilow),cut(ihigh),
     +                   cdfval,powr)
      endif
c
c FINISHED THE MIDDLE,  ARE WE IN THE UPPER TAIL?
c
      if(ipart.eq.2) then
          if(utail.eq.1) then
            powr = 1.0d0
            zval = powint(cdf(ncut),1.d0,cut(ncut),zmax,cdfval,powr)
          else if(utail.eq.2) then
c
c Power interpolation to upper limit "utpar"?
c
            cpow = 1.0d0 / utpar
            zval = powint(cdf(ncut),1.d0,cut(ncut),zmax,cdfval,cpow)
c
c Fit a Hyperbolic Distribution?
c
          else if(utail.eq.4) then
c
c Figure out "lambda" and required info:
c
            lambda = (cut(ncut)**utpar)*(1.0d0-cdf(ncut))
            zval = (lambda/(1.0d0-cdfval))**(1.0d0/utpar)
          else
c
c Error situation - unacceptable option:
c
            ierr = 2
            return
          endif
      endif
      if(zval.gt.zmax) zval = zmax
c
c All finished - return:
c
      return
      end



      real*8 function powint(xlow,xhigh,ylow,yhigh,xval,pow)
c-----------------------------------------------------------------------
c
c Power interpolate the value of y between (xlow,ylow) and (xhigh,yhigh)  for a value of x and a power pow.
c
c-----------------------------------------------------------------------
      implicit real*8(a-h,o-z)
      parameter(EPSLON=1.0d-20)
      powint = ylow +       (yhigh-ylow)*
     +      (((xval-xlow)/amax1(EPSLON,(xhigh-xlow)))**pow)
      return
```

```
      end


      real*8 function ginv(p,ymean,ystd,itrans)
c-----------------------------------------------------------------------
c
c Computes the inverse of the standard normal cumulative distribution function with a numerical approximation from : Abramovitz,
c M. and Stegun, I., 1972, handbook of mathematical functions, 10th printing, National Bureau of Standards, p. 933.
c
c-----------------------------------------------------------------------
c
c Coefficients of approximation:
c
      implicit real*8(a-h,o-z)
      data c0/2.515517d0/,c1/.802853d0/,c2/.010328d0/
      data d1/1.432788d0/,d2/.189269d0/,d3/.001308d0/
c
c Values for + and - infinity:
c
      data gneg/-5.0d0/,gpos/5.0d0/
c
c Check for probability = 0 or 1
c
      if(p.le.0.0d0) then
          ginv = gneg
      else if(p.ge.1.0d0) then
          ginv = gpos
c
c Approximate the function:
c
      else
          pp = p
          if(pp.ge.0.5d0) pp = 1.0d0 - pp
          t   = dsqrt(dlog(1.d0/(pp*pp)))
          t2  = t*t
          t3  = t2*t
          ginv = t - (c0+c1*t+c2*t2)/(1+d1*t+d2*t2+d3*t3)
          if(p.eq.pp) ginv=-ginv
      endif
      if (itrans.ge.1) then
        ginv=dexp(ystd*ginv+ymean)
      end if
c
c Return with ginv:
c
      return
      end


      subroutine getlag
c-----------------------------------------------------------------------
c           Establish the number and location of the lags to consider
c     *********************************************************
c
c
c
c Author: C.V. Deutsch                    Date: April 1992
c-----------------------------------------------------------------------
      include  'metro.inc'
      real*8    maxcov
c     real*8    dist(maxlag)
      logical   covaf
c
c Compute maximum covariance:
c
      covaf  = .true.
```

```
      maxcov = cova3(0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,0.0d0,nst,c0,it,
     +           cmax,cc,aa,ang1,ang2,ang3,anis1,anis2,covaf)
      covaf = .false.
c
c Initialize the variogram and lag arrays:
c
      do 1 i=1,nlag
         varmod(i) = 1.0d+20
c         dist(i)  = 1.0d10
         ixl(i)   = 0
         iyl(i)   = 0
         izl(i)   = 0
 1    continue
c
c Calculate the Experimental Variogram:
c

      na  = 0
      nxl = nx/2
      nyl = ny/2
      nzl = nz/2
      do 20 ix=0,nxl
      do 20 iy=-nyl,nyl
      do 20 iz=-nzl,nzl
         if(ix.eq.0.and.iy.eq.0.and.iz.eq.0) go to 2
         if(ix.eq.0.and.iy.le.0.and.iz.le.0) go to 2
         dx = dble(ix) * xsiz
         dy = dble(iy) * ysiz
         dz = dble(iz) * zsiz
c         dxyz=sqdist(0.d0,0.d0,0.d0,dx,dy,dz,ang1,ang2,
c     +ang3,anis1,anis2)
         vario = maxcov - cova3(0.0d0,0.0d0,0.0d0,dx,dy,dz,nst,c0,it,
     +                   cmax,cc,aa,ang1,ang2,ang3,anis1,
     +                   anis2,covaf)

         if(na.eq.nlag.and.vario.gt.varmod(na)) go to 2
c          if(na.eq.nlag.and.dxyz.gt.dist(na)) goto 2
c
c Consider this sample (it will be added in the correct location):
c
         if(na.lt.nlag) na = na + 1
c         do no=1,na
c           if (vario.eq.varmod(no)) goto 20
c           if (dxyz.eq.dist(no))goto 20
c         end do
         ixl(na)   = ix
         iyl(na)   = iy
         izl(na)   = iz
         varmod(na) = vario
c         dist(na)=dxyz
         if(na.eq.1) go to 2
c
c Sort samples found thus far in increasing order of distance:
c
         n1 = na-1
         do 3 ii=1,n1
            k=ii
            if(vario.lt.varmod(ii)) then
c              if(dxyz.lt.dist(ii)) then
               jk = 0
               do 4 jj=k,n1
                  j  = n1-jk
                  jk = jk+1
                  j1 = j+1
                  varmod(j1) = varmod(j)
c                  dist(j1)=dist(j)
                  ixl(j1)   = ixl(j)
                  iyl(j1)   = iyl(j)
```

```
                  izl(j1)   = izl(j)
 4                continue
                  varmod(k) = vario
c                  dist(k)=dxyz
                  ixl(k)   = ix
                  iyl(k)   = iy
                  izl(k)   = iz
                  go to 2
               endif
 3          continue
c
c 1/22/97 : The following debugging information are added by Tai-Sheng Liou to see how lags are chosen
c

 2     if (na.eq.nlag.and.vario.gt.varmod(na))goto 20
c 2     if (na.eq.nlag.and.dxyz.gt.dist(na)) goto 20

 20    continue
c      minus=1

c      do i=1,nlag
c        iyl(i)=iyl(i)*(-1)**minus
c        minus=minus+1
c        dx=ixl(i)*xsiz
c        dy=iyl(i)*ysiz
c        dz=izl(i)*zsiz
c        varmod(i)=maxcov-cova3(0.0d0,0.0d0,0.0d0,dx,dy,dz,nst,c0,it,
c     +                  cmax,cc,aa,ang1,ang2,ang3,anis1,
c     +                  anis2,covaf)
c      end do

c
c Debugging information:
c
      write(ldbg,100) nlag
 100  format(/'Closest ',i3,' lags:  Lag number  variogram   offsets')

      do 10 i=1,nlag
         write(ldbg,101) i,varmod(i),ixl(i),iyl(i),izl(i)
c
c 1/22/97 : The following line is added by Tai-Sheng Liou

c          write(7788,101) i,varmod(i),ixl(i),iyl(i),izl(i)
c
 101       format(i2,1x,f12.4,3i3)
 10   continue
c
c Return with the closest lags:
c
      return
      end



      subroutine gridxyz(iseed)
c-----------------------------------------------------------------------------------------------------------------------------------------------
c
c This subroutine generate a sample that honors the input mean and variance of the variable on the non-conditioning points. The
c probability distribution of the variable is assumed to be log-normal.
c
c Input :
c       ymean : Mean of ln(x) [y=ln(x)]
c       ystd  : S.T.D. of ln(x)
c       iseed : Initial seed number
c
c Output :
c       var(nx,ny,nz) : Image of the random field in the grid
c
```

```
c Remark :
c       noncond = # of data points that are not conditioning data
c
c-----------------------------------------------------------------------------------------------------------------------------

      include 'metro.inc'
      real*8 xtry(maxx,maxy,maxz),xsave(50),pasp(50)
      logical image
c
c Generate the initial image in the grid
c (The conditioning data is not changed)
c
      p=getrand(iseed)
      iseed=0
      nxyz=nx*ny*nz
c
c 2/10/97 : Define xcut,i.e., the cutoff value of z
c         (1) itrans=1 : z'=max(z-xcut,xcut0)
c         (2) itrans=2 : z'=max(z,xcut0)
c                   xcut0=minimum PM (or Ks) value
c
c i.e. if itrans=1 and xcut.ne.0, use input value as cutoff PM value
c     if itrans=1 and xcut=0, xcut=LN-1(pcut,ymean,ystd)
c     if itrans=2, the cutoff PM value(xcut) is defaulted as 0.0
c

c
c-----Reset cutoff PM (or Ks) value (xcut) to zero if itrans=2
c     If itrans=1 and xcut=0 => iterate until pasp=ptarget
c     If itrans=1 and xcut>0 => no iterations, transform z'=max(z-xcut,xcut0)
c
      if (itrans.eq.2) xcut=0.0
      cutsave=xcut
      write(*,20) xcut
      write(*,*)'************************************************'
20    format(' Initial PM cutoff       = ',f7.4)

c
c Check again the type of simulation:
c
      write(*,*)
      write(ldbg,*)
      if(igauss.eq.0) then
        write(*,*) 'You are using a non-parametric distribution ...'
        write(ldbg,*) 'You are using a non-parametric distribution ...'
      elseif (igauss.eq.1) then
        inquire(file=imageinfl,exist=image)
        if (.not.image) then
        write(*,*) 'You are simulating standard normal deviates ...'
        write(ldbg,*) 'You are simulating standard normal deviates ...'
        end if
      end if

c
c 2/13/97 : Define the value of asperity contact.
c (a) The lowest possible of LN variate, i.e., ginv(0.0,...)
c (b) A very small value defined by user, e.g., 1.0e-4
c
c 7/14/97 : Iterate the generating process until the proportion of asperity contact reaches the target proportion, ptarget
c
      xcut0=aspcut
      do i=1,50
        pasp(iter)=0.d0
        xsave(iter)=0.d0
      end do
      iter=1
      psave=0.d0
      pdif=1.d0
```

```
      smean=0.d0
      smeanln=0.d0
c
      if (itrans.eq.2) then
        if(pcut.eq.ptarget)then
         do 17 i=1,nx
         do 17 j=1,ny
         do 17 k=1,nz
           if(.not.cond(i,j,k))then
             p=getrand(iseed)
             var(i,j,k)=ginv(p,ymean,ystd,itrans)
             do while(var(i,j,k).gt.zmax.or.var(i,j,k).lt.zmin)
               p=getrand(iseed)
               var(i,j,k)=ginv(p,ymean,ystd,itrans)
             end do
           end if
           smean=smean+var(i,j,k)
17       continue
        else
        nop=nxyz*(1.d0-pcut)
        dowhile(nop.gt.nxyz*(ptarget-pcut))
13        i1 = int(getrand(seed)*nx)+1
          j1 = int(getrand(seed)*ny)+1
          k1=1
          if(var(i1,j1,k1).ne.0.0.or.cond(i1,j1,k1)) go to 13
          p=getrand(iseed)
          var(i1,j1,k1)=ginv(p,ymean,ystd,itrans)

c         do while(var(i1,j1,k1).gt.zmax.or.var(i1,j1,k1).lt.zmin)
c           p=getrand(iseed)
c           var(i1,j1,k1)=ginv(p,ymean,ystd,itrans)
c         end do

          if(var(i1,j1,k1).gt.zmax)var(i1,j1,k1)=zmax
          if(var(i1,j1,k1).lt.zmax)var(i1,j1,k1)=zmin
          smean=smean+var(i1,j1,k1)
          nop=nop-1
        end do
        end if
      else
        if(xcut.eq.0)then
        DOWHILE(pdif.gt. 1.0d-3)
        smean=0.0d0
        smeanln=0.d0
        if(iter.eq.2) then
           xsave(iter)=ginv(pcut,ymean,ystd,itrans)
        elseif(iter.gt.2) then
           xsave(iter)=(ptarget-pasp(1))*xsave(iter-1)/
     +              (pasp(iter-1)-pasp(1))
        end if
        if(iter.ge.2)xcut=xsave(iter)
        do 1 i=1,nx
        do 1 j=1,ny
        do 1 k=1,nz
          p=getrand(iseed)
          if (.not.cond(i,j,k)) then
           xtry(i,j,k)=ginv(p,ymean,ystd,itrans)
           var(i,j,k)=amax1(xtry(i,j,k)-xcut,xcut0)
           if(var(i,j,k).gt.zmax)var(i,j,k)=zmax
           if(var(i,j,k).lt.zmin)var(i,j,k)=zmin
          else
            var(i,j,k)=xcut0
          end if
          if(var(i,j,k).eq.xcut0) pasp(iter)=pasp(iter)+1.d0
          smean=smean+var(i,j,k)
          if(var(i,j,k).gt.0.d0)smeanln=smeanln+dlog(var(i,j,k))
1       continue
        pasp(iter)=pasp(iter)/dble(nxyz)
```

```
      write(*,3) iter,xcut,ptarget,pasp(iter)
      paspsave=pasp(iter)
3      format(' iter=',i2,' cut=',f7.4,' ptarget=',f6.2,' pasp=',
     + e20.9)
      psave=pasp(iter)
      iter=iter+1
      pdif=abs(ptarget-psave)
      if (ptarget.eq.0.d0) pdif=0.d0
      END DO
      else
      do 111 i=1,nx
      do 111 j=1,ny
      do 111 k=1,nz
        p=getrand(iseed)
        if (.not.cond(i,j,k)) then
          xtry(i,j,k)=ginv(p,ymean,ystd,itrans)
          var(i,j,k)=amax1(xtry(i,j,k)-xcut,xcut0)
          if(var(i,j,k).gt.zmax)var(i,j,k)=zmax
          if(var(i,j,k).lt.zmin)var(i,j,k)=zmin
        else
          var(i,j,k)=xcut0
        end if
        if(var(i,j,k).eq.xcut0) pasp(iter)=pasp(iter)+1.d0
        smean=smean+var(i,j,k)
        if(var(i,j,k).gt.0.d0)smeanln=smeanln+dlog(var(i,j,k))
111     continue
      end if
    end if

    smean=smean/dble(nxyz)
    smeanln=smeanln/dble(nxyz)
    sstd=0.0d0
    sstdln=0.d0
    do 2 i=1,nx
    do 2 j=1,ny
    do 2 k=1,nz
      sstd=sstd+(var(i,j,k)-smean)*(var(i,j,k)-smean)
      if(var(i,j,k).gt.0.d0)sstdln=sstdln+(dlog(var(i,j,k))-
     *  smeanln)*(dlog(var(i,j,k))-smeanln)
2    continue
    sstd=dsqrt(sstd/(dble(nxyz)-1.d0))
    sstdln=dsqrt(sstdln/(dble(nxyz)-1.d0))
    write(*,*)
    write(*,60) smeanln,sstdln*sstdln
    write(*,70) smean,sstd*sstd
    write(ldbg,*)
c     write(ldbg,60) smeanlnx,sstdlnx*sstdlnx
    write(ldbg,70) smean,sstd*sstd
    if(imageoutfl(1:10).ne.'nodata.dat') then
      write(limageout,*) 'Initial image data'
      write(limageout,*) 5
      write(limageout,*) 'X location'
      write(limageout,*) 'Y location'
      write(limageout,*) 'Z location'
      write(limageout,*) 'k before the cutoff'
      write(limageout,*) 'k after the cutoff'
c
c change ymn to yini for output using DEPTH as the vertical distance
c
      do 40 i=1,nx
      do 40 j=1,ny
      do 40 k=1,nz
        xx=xmn+xsiz*dble(i-1)
        yy=ymn+ysiz*dble(j-1)
        zz=zmn+zsiz*dble(k-1)
        write(limageout,30) xx,yy,zz,xtry(i,j,k),var(i,j,k)
        if (var(i,j,k).eq.xcut0) then
          if(.not.cond(i,j,k))then
```

```
            write(550,30) xx,yy,zz,0.0
          end if
        end if
40      continue
    end if
    close(limageout)
30  format(3(f6.2,2x),2f10.4)
60  format(' Generated statistics of ln(PM) : mean= ',e12.4,' Var = ',
   +      e12.4)
70  format(' Generated statistics of  PM   : mean= ',e12.4,' Var = ',
   +      e12.4)

    return
    end


    subroutine etapdf(iseed)
c-----------------------------------------------------------------------------------------------------------------------------------------------
c
c This subroutine generates a sample that uses user-specified pdf (CDF) with the following three options:
c
c Case 1: log-normal (see gridxyz above)
c
c Case 2 :(Exponential)
c       f(x)= 0.25 d(x)          x<=0
c       f(x)= lumda*exp(-lumda*x) x>0
c
c Case 3 : step-wise uniform
c       f(x)= 0.25 d(x)  x<=0
c       f(x)= 0.1       0<x<1
c       f(x)= 0.0625    1<x=5
c       f(x)= 0.02      5<x<10
c       f(x)= 0.01     10<x<=25
c       f(x)= 0.004    25<x<=50
c       f(x)= 0.001    50<x<=100
c
c-----------------------------------------------------------------------------------------------------------------------------------------------

    include 'metro.inc'
    real*8 xtry(maxx,maxy,maxz),xsave(50),pasp(50)
    real*8 lumbda
c
c Generate the initial image in the grid
c (The conditioning data is not changed)
c
    smean=0.0d0
    p=getrand(iseed)
    iseed=0
    nxyz=nx*ny*nz
c
c Set non-conditioning data to 0.000
c
    nocond=0
    do 3 i=1,nx
    do 3 j=1,ny
    do 3 k=1,nz
      if (.not.cond(i,j,k)) var(i,j,k)=0.0d0
      if (cond(i,j,k))nocond=nocond+1
3   continue

c
c Check again the type of simulation:
c
    write(*,*)
    write(ldbg,*)
    if(igauss.gt.2) then
      icase=igauss-1
      write(*,*) 'You are using ',icase,' -th kind of CDF'
```

```
      write(ldbg,*) 'You are using ',icase,' -th kind of CDF'
    end if
c
c 6/30/97 : Generate initial permeability field using
c       user-defined CDF
c
    nop=0
    nopmax=10000*(1.d0-ptarget)
    icase=igauss-1
    if(icase.eq.2)then
c
c Case 2 -- 25% exponentially decayed pdf with
c
c      p(x)=lumda*exp(-lumda*x), CDF(x)=0.25+0.75*(1-exp(-lumbda*x))
c      lumbda=1/(mean of PM)


      lumbda=0.065
      write(ldbg,*) 'lumbda=',lumbda
      write(ldbg,*) 'nop = ',nop
      write(ldbg,*) 'ptarget=',ptarget
      write(ldbg,*) 'nopmax=',nopmax
      write(ldbg,*) 'xcut0=',xcut0
      if (pcut.eq.ptarget)then
c
c Assign pm values to grid blocks sequentially if fraction of conditioning data is exactly ptarget
c
        do 100 i=1,nx
        do 100 j=1,ny
        do 100 z=1,nz
          if(.not.cond(i,j,k))then
          do while(var(i,j,k).eq.xcut0)
c           .or.var(i,j,k).gt.zmax.or.var(i,j,k).lt.zmin)
            p=getrand(iseed)
            if(p.gt.ptarget)var(i,j,k)=-dlog(1.d0-(p-ptarget)/
   +         (1.d0-ptarget))/lumbda
          end do
          if(var(i,j,k).gt.zmax)var(i,j,k)=zmax
          if(var(i,j,k).lt.zmin)var(i,j,k)=zmin
          end if
100       continue
      else
c
c Assign pm values to grid blocks randomly if fraction of conditioning data is less than ptarget
c
      do while(nop.lt.nopmax)
6       i=int(getrand(iseed)*nx)+1
        j=int(getrand(iseed)*ny)+1
        k=1
        if(var(i,j,k).ne.xcut0.or.cond(i,j,k))goto 6
        do while(var(i,j,k).eq.xcut0)
c         .or.var(i,j,k).gt.zmax.or.var(i,j,k).lt.zmin)
          p=getrand(iseed)
          if(p.gt.ptarget)var(i,j,k)=-dlog(1.d0-(p-ptarget)/
   +       (1.d0-ptarget))/lumbda
        end do
        if(var(i,j,k).gt.zmax)var(i,j,k)=zmax
        if(var(i,j,k).lt.zmin)var(i,j,k)=zmin
        if(p.gt.ptarget)nop=nop+1
       end do
       write(*,*) 'End of generating sample data'
      end if
c
c Case 3 -- 25% of total asperity contacts
c

     elseif(icase.eq.3)then
      if(pcut.eq.ptarget)then
```

```
c
c Assign pm values to grid blocks sequentially if fraction of conditioning data is exactly ptarget
c
      do 200 i=1,nx
      do 200 j=1,ny
      do 200 k=1,nz
      do while (p.lt.0.25.or.p.gt.1.d0)
       p=getrand(iseed)
       if(p.gt.0.25.and.p.le.0.35)var(i,j,k)=(p-0.25)/0.1d0
       if(p.gt.0.35.and.p.le.0.60)var(i,j,k)=1.+(p-0.35)/6.25d-2
       if(p.gt.0.60.and.p.le.0.70)var(i,j,k)=5.+(p-0.60)/2.d-2
       if(p.gt.0.70.and.p.le.0.85)var(i,j,k)=10.+(p-0.70)/1.d-2
       if(p.gt.0.85.and.p.le.0.95)var(i,j,k)=25.+(p-0.85)/4.0d-3
       if(p.gt.0.95.and.p.le.1.00)var(i,j,k)=50.+(p-0.95)/1.d-3
      end do
200    continue
      else
c
c Assign pm values to grid blocks randomly if fraction of conditioning data is less than ptarget
c
     do while (nop.lt.nopmax)
7      i = int(getrand(seed)*nx)+1
       j = int(getrand(seed)*ny)+1
       k = 1
       if(var(i,j,k).ne.0.0.or.cond(i,j,k)) go to 7
       p=getrand(iseed)
        if(p.gt.0.25.and.p.le.0.35)var(i,j,k)=(p-0.25)/0.1d0
        if(p.gt.0.35.and.p.le.0.60)var(i,j,k)=1.+(p-0.35)/6.25d-2
        if(p.gt.0.60.and.p.le.0.70)var(i,j,k)=5.+(p-0.60)/2.d-2
        if(p.gt.0.70.and.p.le.0.85)var(i,j,k)=10.+(p-0.70)/1.d-2
        if(p.gt.0.85.and.p.le.0.95)var(i,j,k)=25.+(p-0.85)/4.0d-3
        if(p.gt.0.95.and.p.le.1.00)var(i,j,k)=50.+(p-0.95)/1.d-3
        if(p.gt.0.25.and.p.le.1.0) nop=nop+1
c       elseif(icase.eq.4) then
c
c Case 4-- 30% of total asperity contacts
c
c         if(p.gt.0.25.and.p.le.0.30)var(i,j,k)=(p-0.25)/0.05d0
c         if(p.gt.0.30.and.p.le.0.40)var(i,j,k)=1.+(p-0.30)/2.5d-2
c         if(p.gt.0.40.and.p.le.0.45)var(i,j,k)=5.+(p-0.40)/1.d-2
c         if(p.gt.0.45.and.p.le.0.55)var(i,j,k)=10.+(p-0.45)/6.66667d-3
c         if(p.gt.0.55.and.p.le.0.65)var(i,j,k)=25.+(p-0.55)/4.0d-3
c         if(p.gt.0.65.and.p.le.0.75)var(i,j,k)=50+(p-0.65)/4.d-3
c         if(p.gt.0.75.and.p.le.1.00)var(i,j,k)=75.+(p-0.75)/1.d-2
c         if(p.gt.0.25.and.p.le.1.0) nop=nop+1
     end do
     end if
     end if

     smean=0.0
     do 5 i=1,nx
     do 5 j=1,ny
     do 5 k=1,nz
        smean=smean+var(i,j,k)
5    continue
10   write(*,*)
     write(*,*) 'nop = ',nop
     smean=smean/dble(nxyz)
     sstd=0.0d0
     do 2 i=1,nx
     do 2 j=1,ny
     do 2 k=1,nz
        sstd=sstd+(var(i,j,k)-smean)*(var(i,j,k)-smean)
2    continue
     sstd=sqrt(sstd/(dble(nxyz)-1.d0))
     write(*,*)
     write(ldbg,*)
     write(*,70) smean,sstd*sstd
```

```
      write(ldbg,70) smean,sstd*sstd
c
c change ymn to yini for output using DEPTH as the vertical distance
c
c     yinit=-(ysiz*dble(ny)-ysiz/2.d0)
      if(imageoutfl(1:10).ne.'nodata.dat') then
        write(limageout,*) 'Initial image data'
        write(limageout,*) 5
        write(limageout,*) 'X location'
        write(limageout,*) 'Y location'
        write(limageout,*) 'Z location'
        write(limageout,*) 'k before the cutoff'
        write(limageout,*) 'k after the cutoff'
        do 40 i=1,nx
        do 40 j=1,ny
        do 40 k=1,nz
          xx=xmn+xsiz*dble(i-1)
          yy=ymn+ysiz*dble(j-1)
          zz=zmn+zsiz*dble(k-1)
          write(limageout,30) xx,yy,zz,var(i,j,k),0.0
          if (var(i,j,k).eq.0.d0) then
            if(.not.cond(i,j,k))then
c             yy=yinit+ysiz*dble(j-1)
              write(550,30) xx,yy,zz,0.0
            end if
          end if
40      continue
      end if
      close(limageout)
30    format(3(f6.2,2x),2f10.4)
60    format(' Generated statistics of ln(PM) : mean= ',f7.3,' Var = ',
     +      f11.3)
70    format(' Generated statistics of  PM   : mean= ',f7.3,' Var = ',
     +      f11.3)

      return
      end
```

(3) Sample input file

```
Parameters for SASIM
********************

START OF PARAMETERS:
nodata.dat                      \input initial image file
image.dat                       \output initial image file
asperity.dat                    \conditioning data (if any)
1  2  0  3                      \columns: x,y,z,vr
-1.0e21   1.0e21                \data trimming limits 0=non parametric; 1=Gaussian; 2=left open; 3=Exp; 4=step-wise uniform)
nodata.dat                      \non parametric distribution
3  0                            \columns: vr,wt
0.0   100.0                     \minimum and maximum data values
1     1.0                       \lower tail option and parameter
4     2.0                       \upper tail option and parameter
case.dat                        \output File for simulation
var.dat                         \output File for variogram
varh.dat                        \output file for hori. variogram
varv.dat                        \output file for ver. variogram
3    5000000  25000000          \debug level, reporting interval
case.dbg                        \output file for debugging
nodata.dat                      \output file for lag information
                                \annealing schedule? (0-3 with 0=user supplied, 1=default, 2=fast, 3=very fast)
1.0 0.9 50.0 5.0  3 1.0d-7      \manual schedule:  t0,lambda,ka,k,e,Omin
1                               \1 or 2 part objective function
112063                          \random number seed
1                               \number of simulations
100  0.1   0.2                  \nx,xmn,xsiz
100 -19.9   0.2                 \ny,ymn,ysiz
1   0.5   1.0                   \nz,zmn,zsiz
14  3                           \max lags for conditioning
1   0.0  0                     \nst, nugget, (1=renormalize)
2   0.2  190.0                  \it,aa,cc:    STRUCTURE 1
0.0  0.0  0.0  1.0  1.0         \ang1, ang2, ang3, anis1, anis2:
0   4.0                         \iedge(1:yes, 0:no),wedge
1   1.0                         \icond(1:yes, 0:no),wcond
1.0  1.5   1                    \mlnk,stdlnk,itrans
0.10 0.0  0.0  0.25             \pcut ,aspcut(xcut0), xcut, ptarget
0                               \0:standard, 1:modified Metropolis algorithm
2                               \1:first hbhd, 2:second nbhd
                                \imod (different perturbation mechanisms)
                                \ifield (1(0)=do(not) generate evolving files of PM and semi-variogram fields)
```

Notes:
itrans=0 not transform to LN data, itrans>0, transform to LN data
itrans=1 z'=max(z-PMCUT,aspcut) where PMCUT=xcut, if xcut .ne. 0 or PMCUT=LN-1(ymean,ystd,pcut) if xcut=0
itrans=2 z'=max(z,aspcut), i.e., PMCUT=0 (without shifting)
aspcut=PM value of asperity contact
xcut=cutoff permeability (0=defualt)
pcut=cutoff probability=ratio of asperity contact to total rock volume and is used to calculate xcut

(4) Sample output file

```
Permeability field from simulated annealing
        4
X location NX= 100  DX=  0.10 DX=  0.20
Y location NY= 100  DY=-19.90 DY=  0.20
Z location NZ=  1    DZ=  0.50   DZ=  1.00
Permeability field
0.1000  -19.9000   0.5000     36.1209
0.7000  -19.9000   0.5000      2.7844
.
  .
  .
19.7000   -0.1000   0.5000      0.0000
19.9000   -0.1000   0.5000      2.7400
```

Appendix D. Calculation of effective permeability

(A) Horizontal effective permeability

To calculate the horizontal effective permeability, attach two boundary elements, 'rhs 1' and 'lhs 1' to the entire right and left hand side boundaries, respectively, and let the element 'lhs 1' be inactive such that primary variables in that element will not be changed during simulation. Inject water from 'rhs 1' and turn the gravity off. Run the simulation under single-phase conditions then the pressure in element 'rhs 1' will increase and reach to a constant value after a short transient, i.e., the steady state is reached. Calculate the horizontal effective permeability according to Darcy's law, i.e.,

$$k_{\text{eff,h}} = \frac{\mu q}{\rho A \nabla P} = \frac{\mu q}{\rho A\left(\Delta P / \Delta x\right)} \tag{D-1}$$

where μ [Pa·s] is water viscosity, q [kg/s] is injection rate, ρ [kg/m^3] is water density, A [m^2] is the contact area of 'rhs 1' to the entire right hand side boundary, and ΔP [Pa] and Δx [m] are the pressure difference and horizontal distance between 'rhs 1' and 'lhs 1', respectively.

(B) Vertical effective permeability

Vertical effective permeability can be obtained by following the same procedures as in (A) but attaching two boundary elements, 'top 1' and 'bot 1', to the entire top and bottom boundaries, respectively. Let the element 'bot 1' be inactive and inject water from 'top 1'. Gravity is sill turned off in this case. Therefore, the vertical effective permeability is calculated according to

$$k_{\text{eff,v}} = \frac{\mu q}{\rho A \nabla P} = \frac{\mu q}{\rho A\left(\Delta P / \Delta z\right)} \tag{D-2}$$

where μ, q, and ρ are the same as in (A), A is the contact area of the element 'top 1' to the entire top boundary, ΔP is the pressure difference between 'top 1' and 'bot 1', and Δz is the vertical distance between 'top 1' and 'bot 1'.