CRAFIT: A Computer Program for Calibrating Breakthrough Curves of CRAFLUSH, a One-Dimensional Fracture Flow and Transport Model

by Laura Toran[a]

Abstract

This computer note presents a program called CRAFIT, which automates trial and error curve fitting of breakthrough curves for CRAFLUSH, a one-dimensional fracture flow and transport code. CRAFIT is useful for modeling column experiments where the source function is well-known but fracture geometry and matrix are to be characterized. CRAFIT is demonstrated for curve fitting with three unknowns and five unknowns, and found a single best-fit solution for each case.

Introduction

The CRAFIT program automates trial and error curve fitting for a one-dimensional fracture flow and transport code, CRAFLUSH. CRAFLUSH models flow in parallel plate fractures with matrix diffusion based on the equation described in Sudicky and Frind (1982). The user inputs fracture geometry (spacing and aperture), matrix porosity, tracer characteristics (retardation in the fracture and matrix, decay rate, diffusion rate), and source and background concentrations; the program calculates tracer concentrations in the fracture and in the matrix. CRAFLUSH assumes the matrix surrounding the fracture has a low permeability, so that only diffusion can occur in the matrix, not flow and transport.

Thus, the CRAFLUSH program is useful for modeling settings that can be approximated by one-dimensional flow through parallel fractures. For example, a first approximation of fracture flow in the field might use CRAFLUSH (e.g., McKay et al. 1997). Field characterization is difficult, so characterization of fractures in the laboratory may be useful to develop better techniques. CRAFLUSH is well-suited to modeling column experiments in a one-dimensional flow field (e.g., Cumbie and McKay 1999; Jorgenson et al. 1998; Koch et al. 1999) to estimate fracture parameters.

The CRAFIT program is useful for fitting the breakthrough curves of column experiments where the source function is well-known but fracture geometry and matrix are to be characterized. Specifically, the program can solve for matrix porosity, fracture spacing, fracture aperture, fracture retardation, and matrix retardation given a breakthrough curve of concentrations through time. It can be difficult to fit a curve to five unknowns, and CRAFIT helps the user vary the parameters efficiently. In addition, there is ambiguity in distinguishing tracer behavior in fracture transport because matrix diffusion can produce curves similar to tracers that are retarded (Maloszewski and Zuber 1993). The CRAFIT program helps the user evaluate the possibility of multiple solutions for the same dataset.

Program Description

CRAFIT consists of several subroutines that create a series of CRAFLUSH files, run CRAFLUSH, then calculate a measure of error for each modeled run (Table 1). To create the CRAFLUSH files, the first step is to read a default set of parameters used in all of the runs. These fixed parameters include the diffusion coefficient, the decay constant, and the initial concentration. These data are read from a file called start.cra. The format of the data for start.cra is the same as the input for CRAFLUSH. The parameters input for CRAFLUSH are given in comment statements in the input file. The constant parameters also include the details of the column experiment: injection rate, sample time, and column length. One additional line is added at the top of the start.cra file (not in the original CRAFLUSH input) to provide information about the column experiment. This additional information about the column experiment is used to calculate the fracture velocity from aperture and spacing. Typically, CRAFLUSH takes velocity as input, but assumes the user has calculated the fracture velocity from known parameters. CRAFIT automates calculation of velocity because velocity is fixed by the injection rate. The conversion of injection rate to velocity is described in the program comments. Note that for application of CRAFIT to field scale problems instead of column experiments, a different velocity calculation would be needed.

Random values are generated for five parameters: fracture aperture, fracture spacing, fracture retardation, matrix retardation, and matrix porosity. CRAFIT uses a Latin hypercube sampling scheme (Morris 1998) to vary these parameters. The user selects the range in parameters and Latin hypercube sampling generates a set of runs for Monte Carlo analysis. The Latin hypercube scheme is an efficient method and can also be used to conduct a sensitivity analysis using multiple regression techniques to determine the most sensitive parameters (Toran et al. 1995). The Latin hypercube

---

[a]Temple University, Department of Geology, Philadelphia, PA 19122; ltoran@nimbus.temple.edu

Received April 1999, accepted November 1999.

Table 1
List of Subroutines and Input Files

Subroutine/ Program Name	Purpose
Main	Use Latin hypercube sampling to create random input parameters for spacing, aperture, and porosity
Crafwrit	Reads default parameters (from start.cra) then writes a new CRAFLUSH file using random input parameters
CRAFLUSH	Runs CRAFLUSH, one-dimensional flow and transport in fractures
Findmin	Finds the minimum distance between user-defined interpolation points and modeled breakthrough curves. Sums the errors.
Input File Name	
Lhs.input	Input file for Latin hypercube sampling that provides ranges in parameters, number of runs
Lhs.seeds	Random numbers to seed the Latin hypercube sampling. Overwritten at the end of each calibration sequence.
Start.cra	Initial data for CRAFLUSH input. Provides default values not overwritten by Latin hypercube program.
interp.data	User-defined interpolation data to describe breakthrough curve for calibration.
Scratch File Name	
Craf.in	CRAFLUSH input file created for each Latin run. Overwrites for each run.
Conc.out	Output of concentrations generated by CRAFLUSH, used to compare with interpolation data. Overwritten for each run.
Output File Name	
Lhs.output	Table of the random parameter values generated by Latin hypercube sampling
Err.sum	Summary of the errors by run number. The run number will match the listing order in lhs.output.

scheme divides the range for each parameter into the number of runs (determined by the user), selects a parameter value from each of these divisions, then combines it randomly with the other parameters. A random number generator is initiated with a seed file, and a new seed file is generated each time the Latin hypercube sampling subroutine runs. This seed file can be overwritten or saved to reproduce the same random runs. In the Latin hypercube sampling, no parameter value is selected more than once. The entire range of parameter space is covered, but not all possible combinations of the parameters. If the user selects a large number of runs, there is a small variation in adjacent parameters, and the parameter space is fairly completely covered. For example (Figure. 1), with 1000 runs, the sample space was densely covered for possible combinations for spacing (ranging from 0.1 to 4 cm) and aperture (ranging from 0.001 to 0.2 cm). Although the overlap of large aperture and small spacing is unrealistic, it is needed to obtain cases of large aperture and large spacing. The fitted results should always be checked and plotted by the user to eliminate unrealistic cases.

Once the parameter values are randomly generated, the program writes a new CRAFLUSH input file and runs CRAFLUSH. A breakthrough curve is generated at 150 points. This should be enough to thoroughly characterize the breakthrough curve, but if not, the user should modify the dimension statement in the MAIN subroutine to calculate additional points.

The next subroutine compares the modeled breakthrough curve to an observed breakthrough curve. The user selects up to 25 interpolation points to compare to the modeled curve. The interpolation points are provided by the user in a file called interp.data.

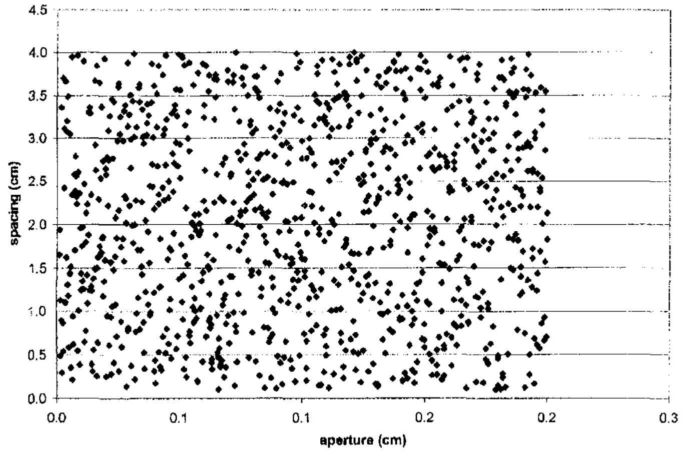

Figure 1. Combinations of values for aperture and spacing for 1000 runs randomly generated by the Latin hypercube sampling.

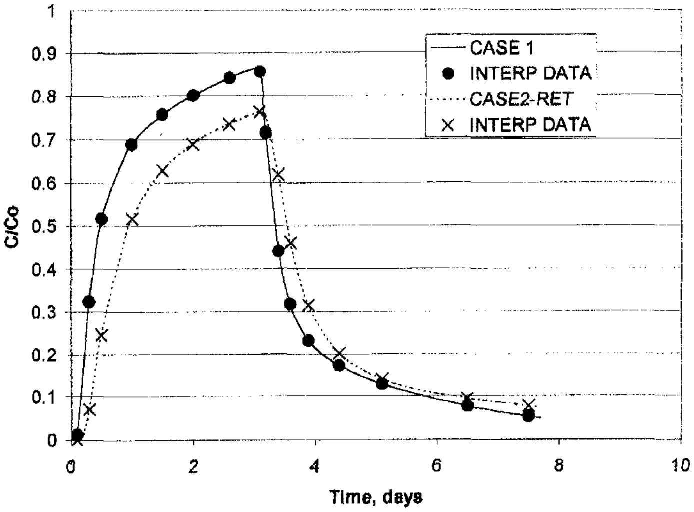

Figure 2. Smoothed breakthrough curve for each synthetic test case with 16 interpolation points to be used for comparison with modeled breakthrough curves.

These interpolation points represent a smoothed breakthrough curve that fits experimental data (Figure 2). The actual experimental data are typically not used in the calibration for two reasons. First, the curve fitting procedure is meant to mimic visual comparison of the shapes of measured and modeled breakthrough curves. Noise in real world data could bias the evaluation of error by adding large error at just one point. Uneven sampling (e.g., sparse data) could bias one portion of the curve instead of fitting an overall shape. Using the interp.data file defined by the user smooths and evenly distributes the data for comparison. Second, the user may want to select points only in one portion of the breakthrough curve such as the rise or the tail. The user has flexibility in locating the interpolation points, but caution is needed to avoid selecting the points with unintentional bias. The user also has the option of using measured data in the interp.data file.

The measure of error calculated by the CRAFIT program is the absolute value of the minimum (orthogonal) distance between the interpolation points and the modeled breakthrough curve. The errors are summed for all interpolation points, then divided by the number of interpolation points. The measure of fit is an average error sum for all of the interpolation points. Other measures of error could be used. Essentially, the technique is similar to calibration by visual comparison. It doesn't make use of optimization or mini-

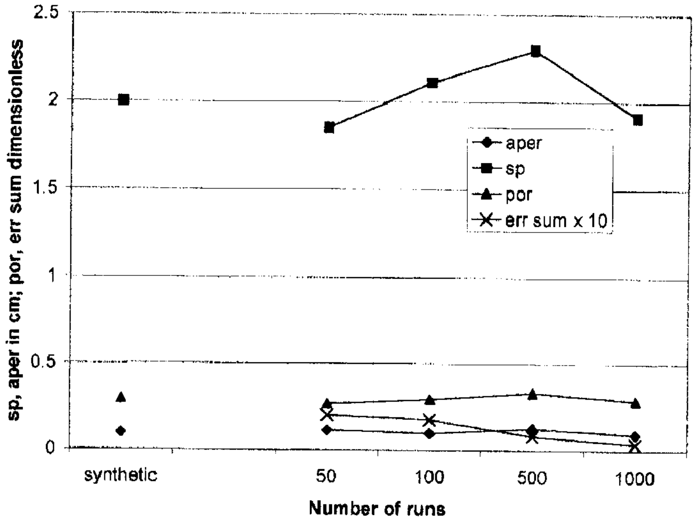

Figure 3. Comparison of best fit model (aperture, spacing, and porosity) to synthetic data for different number of runs. Error summary (times 10) for the best fit also shown.

mization of error, which might not find multiple solutions. The advantages of the calibration technique are that it maps all of the parameter space, and it provides a user-defined error measurement criteria. Again, the user is cautioned not to introduce unintentional bias.

The output of the program is a list of the error sums (err.sum) and parameter values (lhs.output) for each run. The user can rank the error sums and compare the values of the parameters for the lowest error sums.

Example Problems

The CRAFIT program was tested on two synthetic data sets. The problems represented a column experiment in which a conservative tracer was injected for three days at a rate of 4 mL/min, and monitored for seven days. The column was 45 cm long and had 12.5 cm diameter. For each problem the range in parameters used for fitting was the same. The possible range in fracture spacing was 0.1 to 4 cm. The aperture ranged from 0.001 to 0.2 cm. The porosity ranged from 0.1 to 0.5. These parameter values represented a shallow weathered sedimentary rock zone. There was no retardation in the first example, so the range in fracture and matrix retardation were set from 1.0 to 1.0 (no variation). For the second example both retardation factors were allowed to vary from 1.0 to 5.0.

The breakthrough curves for these two examples each have features that can be difficult to fit with trial and error calibration. The breakthrough curves rise steeply and rapidly. Shortly after three days, the concentration falls steeply, but then there is a long tail at about 5% of the initial concentration that persists until the end of monitoring (Figure 2). The example with retardation does not reach as high a concentration and the concentration in the tail falls off a little more gradually.

For the first example problem, comparisons were made with different numbers of interpolation points and different numbers of runs. The number of points for an interpolation curve was 9, 12, 16, or 22, spread all along the breakthrough curve. The number of randomly generated runs was 50, 100, 500, or 1000.

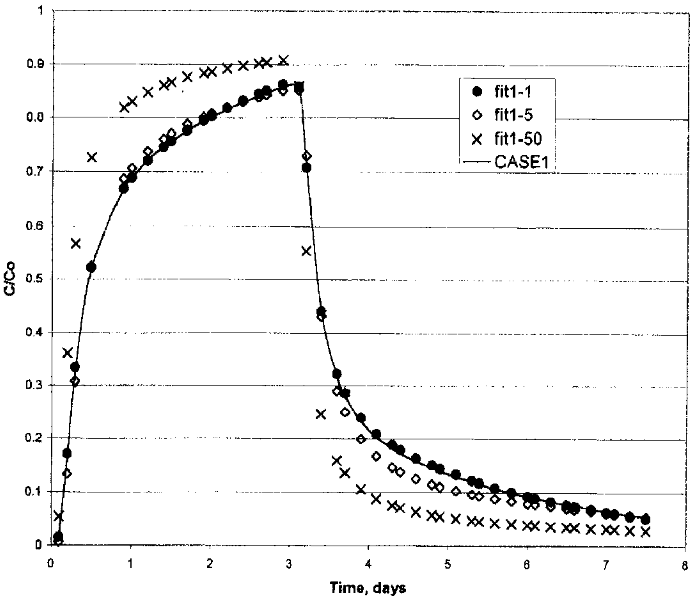

Figure 4. Breakthrough curves for each test case compared with best-fit runs varying three parameters (case 1) or five parameters (case 1 and case 2). The best-fit run varying five parameters for case 1 is not as good as varying only three parameters.

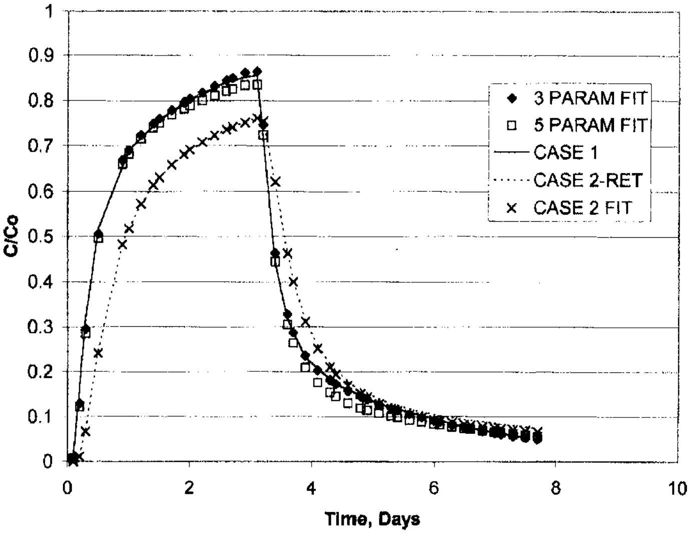

Figure 5. Comparison of breakthrough curves for test case 1 and several randomly generated runs (number indicates ranking of fit: first, fifth, and fiftieth).

Results

For the first example problem the synthetic breakthrough curve could be characterized by 9, 12, 16, or 22 points with the same results. The parameters that showed the best fit were the same for different numbers of interpolation points. In the remaining CRAFIT evaluations, 16 interpolation points were used.

Different numbers of runs showed about the same parameter values for the best fit, although there was some variation (Figure 3). It might also be expected that, with more runs, the sampling could get closer to the correct value. Although the error sum declined with more runs (Figure 3), there was no significant improvement in fit. The spacing was within 0.25 cm of the synthetic value, the aperture within 0.03 cm, and the porosity within 3% for the best fit of each CRAFIT calibration. Some differences are to be expected because the Latin hypercube sampling doesn't create the same combination of parameters each time. Instead, all of the run sizes showed reasonable fits to the synthetic data. It is recommended that the user

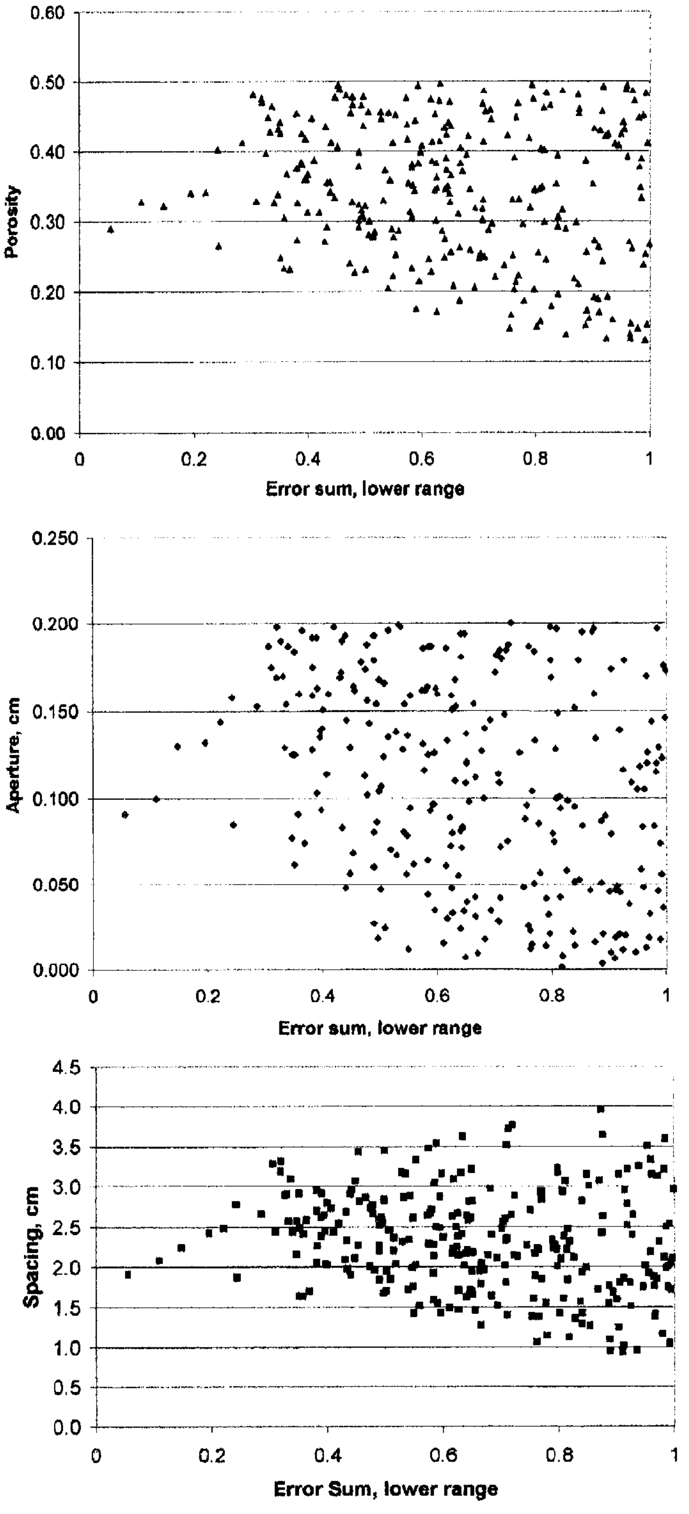

Figure 6. Test case 1 parameter values versus error summary for runs with errors less than one (top 27% of runs). Plots show parameters converge on a single value at low error.

Table 2
Summary of Calibrations Fits for Three Example Cases

	Aper. (cm)	Spac. (cm)	Porosity	Retardation Fracture	Retardation Matrix
Range varied	0.001 – 0.2	0.1 – 0.4	0.1 – 0.5	1.0 – 5.0	1.0 – 5.0
Synthetic case 1	0.1	2.0	0.3	1.0	1.0
Best fit: case 1 3 parameters varied	0.11	2.0	0.29		
Best fit: case 1 5 parameters varied	0.04	1.76	0.19	2.24	2.03
Synthetic case 2	0.1	2.0	0.3	2.0	2.0
Best fit: case 2 5 parameters varied	0.07	2.13	3.04	3.48	2.18

start with a modest number of runs then increase the number to see if an improvement in fit is obtained. A large number of runs samples the parameter space more completely. In the remaining CRAFIT evaluations, 1000 runs were used. Run times on a 266 MHz PC are less than five minutes, so program runtime is not a limiting factor in deciding how many runs to do.

Next, the parameter values of the run with the best fit (lowest error sum) were compared to the parameters of the synthetic test cases (Table 2; Figure 4). When three parameters were varied, the fitted porosity was 29% compared to a synthetic porosity of 30%, the fitted spacing matched the synthetic spacing of 2 cm, and the fitted aperture was 0.11 cm compared to the synthetic aperture of 0.1 cm. These estimates would provide good predictions of tracer behavior.

Next, five parameters were varied for the first test problem 1, with retardation of the fracture and matrix considered unknowns. With five unknowns, CRAFIT did not converge on a solution that was as good a fit as when only three parameters are varied (Table 2; Figure 4). In particular, the retardation factors were around 2 instead of 1. One reason for a poorer fit is that for the same number of runs, fewer are created that are close to the correct solution because the additional parameters make the sampling space larger.

However, in the second test problem, the five parameters all vary, but in this case the synthetic curve has retardation factors are greater than one. For synthetic cases CRAFIT again converges on a single best-fit solution (Table 2; Figure 4). The aperture and retardation of the matrix are not as close to the correct solution as the other parameters, but the solution is not as sensitive to these two parameters (Toran et al. 1995).

There was a difference in match to the breakthrough curve between the run with the lowest error (fit-1) and the runs that are in the top 10 but have larger error sums (Figure 5). This is shown for the case with three parameters. The best-fit run directly overlays the synthetic breakthrough curve. The run with the fifth smallest error (fit1-5) fits well in the rise and the peak, but is about 3% lower in the tail. The fit that ranks fiftieth in error (fit1-50) shows how the fit degrades to a poor fit almost everywhere. It is important to make plots of several of the best runs to see how sensitive the model is to parameters and what portions of the breakthrough curve the model does and does not fit.

The solutions for each of the parameters converged on a single set of parameters rather than multiple possible combinations of the three parameters or five parameters. When the parameter values for test case 1 are plotted against the error sum (i.e., plot scaled to show the better runs), each of the parameters converges on a single point rather than multiple points within the range (Figure 6).

Program Limitations

This program automates trial and error curve fitting. It is not meant to be a substitute for more rigorous minimization methods, but it is a tool for preliminary analysis of curve fitting. CRAFIT does not currently calculate model sensitivity, although the data produced could be analyzed for model sensitivity.

The user can bias the answer depending on the points selected for interpolation data. Sometimes this bias is desired, when the tail or the rise of the breakthrough curve is being investigated separately. The user needs to be careful not to introduce bias unintentionally. The Latin hypercube method provides the same spatial coverage for a parameter with a wide or a narrow range; values will be more tightly spaced for a narrow range than for a wide range. This means that the user should select the number of runs that samples the widest range in parameters completely.

The user is cautioned not to vary retardation for a conservative tracer. When five parameters are allowed to vary, Latin hypercube sampling does not produce as many runs close to the solution and the best fit is not optimized. The user needs to establish which parameters are really unknown and a reasonable range for these unknowns.

Summary and Availability

Fracture flow and transport is characterized by a large number of parameters. It can be difficult for the user to vary all of the possible parameters by trial and error calibration. The CRAFIT program makes it easier to vary a five-parameter problem.

CRAFIT is available free from anonymous ftp:155.247.96.42. Source code, example input files, and executable are available. CRAFLUSH is called as a subroutine for CRAFIT; the CRAFLUSH version is dated 4/92. Ed Sudicky at the University of Waterloo generously made CRAFLUSH available as part of this code.

References

Cumbie, D., and L. McKay. 1999. Influence of diameter on particle transport in a weathered and fractured shale saprolite. *Journal of Contaminant Hydrology* 37, no. 1/2: 139–157.

Jorgensen, P., L. McKay, and N Spliid. 1998. Evaluation of chloride and pesticide transport in a fractured clayey till using large undisturbed column and numerical modeling. *Water Resources Research* 34, no. 4: 539–553.

Koch, E., L. Toran, G. Moline, and K. Firestone. 1999. Modeling effective fractures in a well-characterized intact core. American Geophysical Union Spring Meeting, May 31–June 4, Boston, Massachusetts. *EOS Transactions* 80, no. 17: Abstract #220-05.

Maloszewski, P., and A. Zuber. 1993. Tracer experiments in fissured rocks: Matrix diffusion and the validity of models. *Water Resources Research* 18, no. 8: 645–658.

McKay, L., P. Stafford, and L. Toran. 1997. EPM modeling of a field-scale tritium tracer experiment in fractured, weathered shale. *Ground Water* 35, no. 6: 997–1007.

Morris, M. 1998. Personal communication, January, Oak Ridge National Laboratory, Oak Ridge, Tennessee.

Sudicky, E., and E. Frind. 1982. Contaminant transport in fractured porous media: Analytical solutions for a system of parallel fractures. *Water Resources Research* 18, no. 6: 634–1642.

Toran, L., A. Sjoreen, and M. Morris. 1995. Sensitivity analysis of solute transport in fractured porous media. *Geophys. Res. Let.* 22, 1433–1436.